MICRAL
USERS MANUAL

réalisations
études
électroniques
zone d'activités de courtaboeuf
avenue de scandinavie
91400 orsay
907 47 77

JAN. 74

HARDWARE

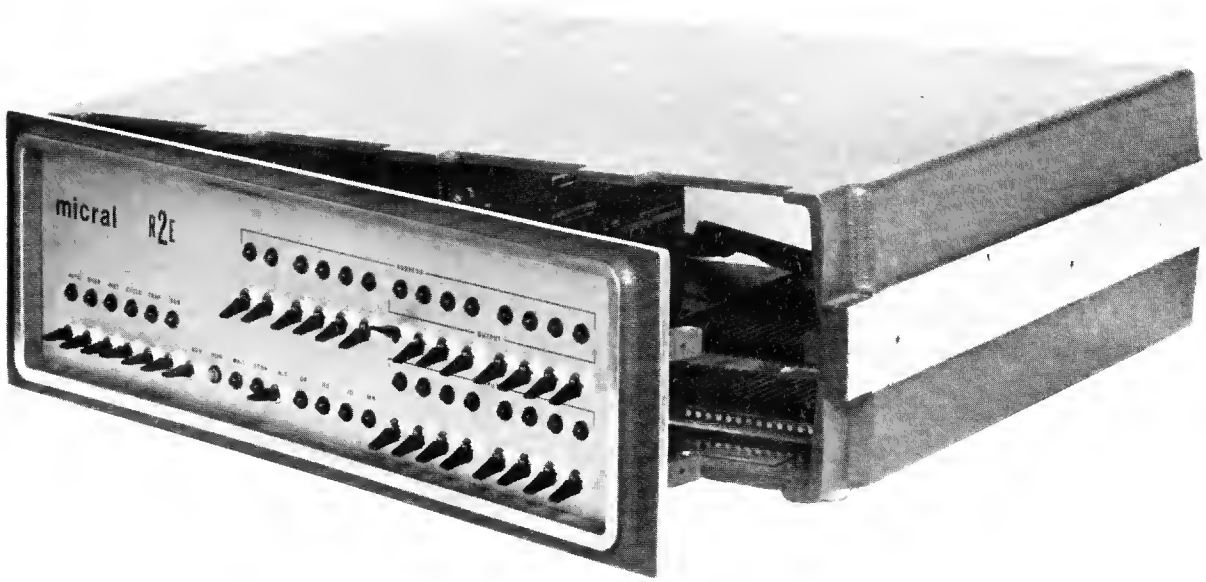


PHOTO 1 MICRAL with front control panel

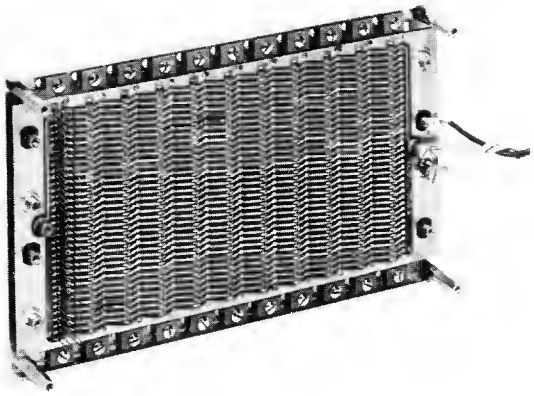


PHOTO 2 PLURIBUS

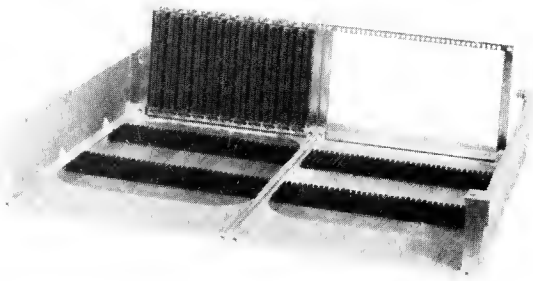


PHOTO 3 Inside view of the MICRAL with one Pluribus

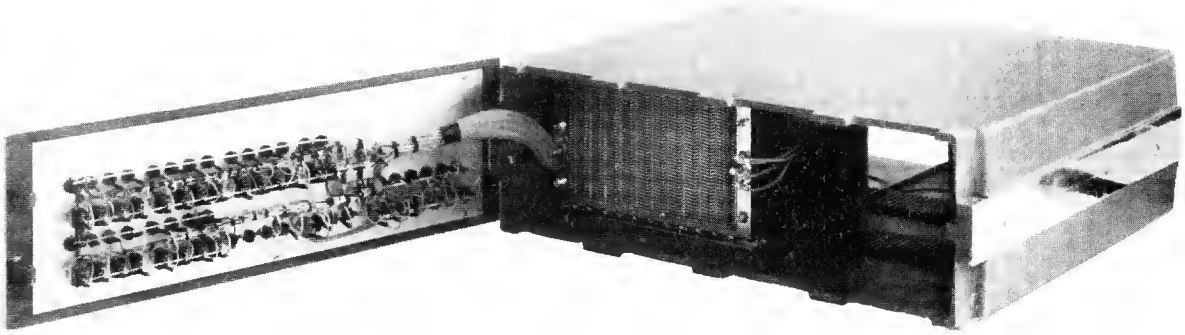


PHOTO 4 MICRAL with front panel open

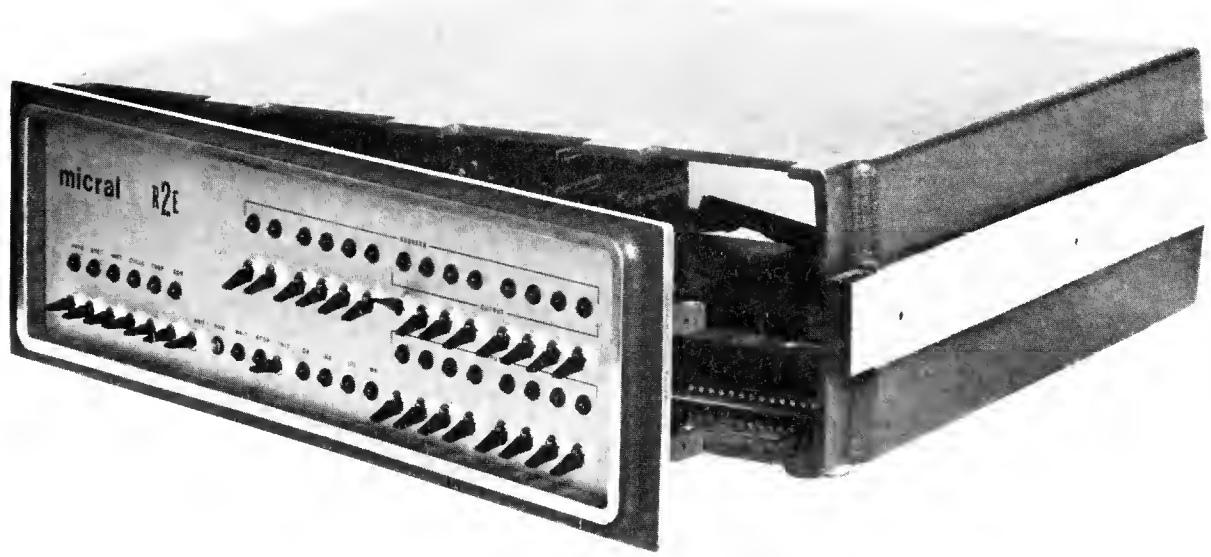


PHOTO 1.1 Computer block with front pannel

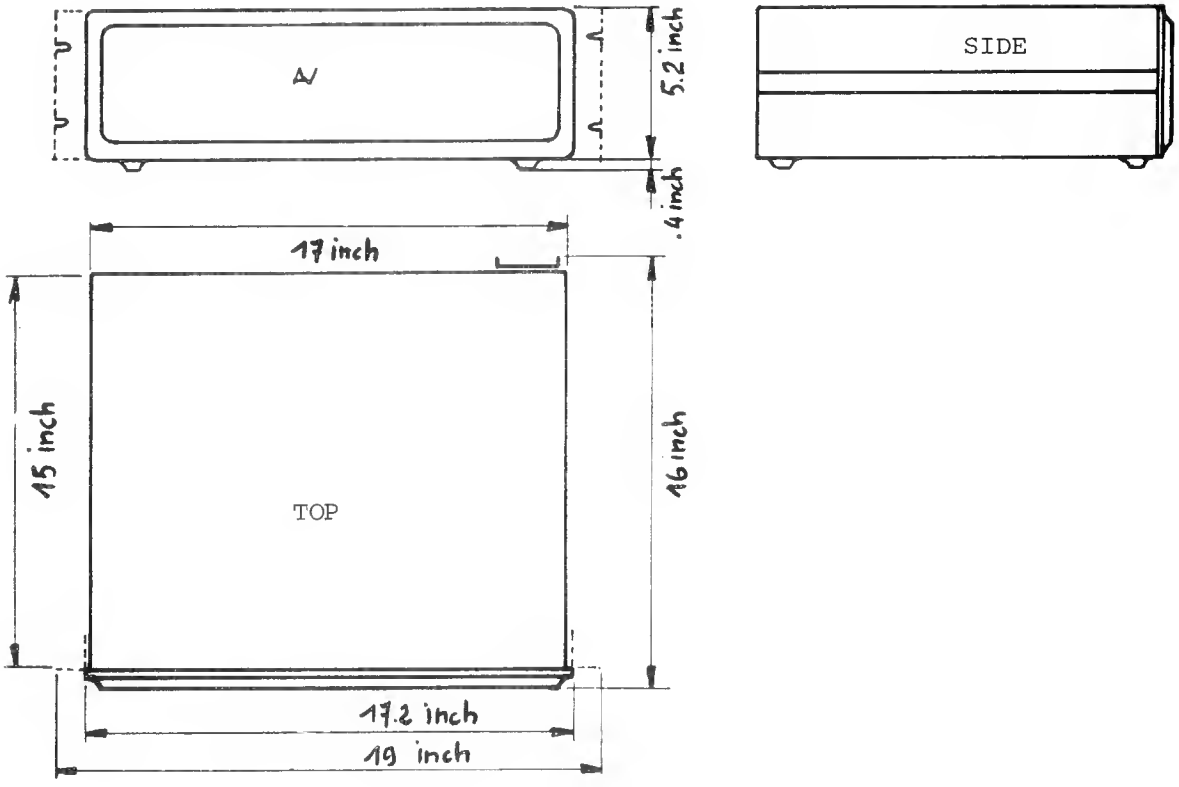


Figure 1.1 Computer block size

I - GENERAL INFORMATION

I.1. Description

MICRAL has two parts :

- the computer block;

- the power supply block.

I.2. Computer block

The computer block consists of a cast of aluminium frame on which the pluribus is mounted. The latter has a printed circuit plugged into eleven identical 74-pin connectors through which the information is fed into and taken out of the pluribus. The connectors can accept any of the cards making up the MICRAL system at random.

In the standard version, one can load up to eleven cards on the pluribus. However, enough room is left on the frame to add on a second pluribus, thereby providing a maximum capacity of 22 standard cards.

The user can also replace the pluribus by a personalized bus to adapt his own Hardware. Note that all the power supply is oversized so to feed the necessary power to all the standard and special circuits. The outside connections are carried out at the end of the cards. The cards are built so as to prevent their being set upside-down on the pluribus. Picture 1.1. shows the mechanical dimensions of the computer block. The computer block can be supplied in the rack form, upon request.

I.3. Power supply

Picture 1.1 shows MICRAL fitted with both its power supply and computer block. The two blocks are connected in the basic configuration by a 2,5 meters long cable. The main supply cable is 2 m long.

The power supply block includes the start switch as well as two fuses to protect the equipment (one for each phase). The main supply consumption, for the standard configuration, is less than 250 W. One must use a D1 TD 2 type, 2 A delay fuse. The power supply block feeds the installation with the following regulated voltages :

$$5\ V \pm 1\% \text{ at } 10\ A.$$

$$-\ 9\ V \pm 1\% \text{ at } 4\ A.$$

Picture 1.2 shows the mechanical dimensions of the power supply unit.

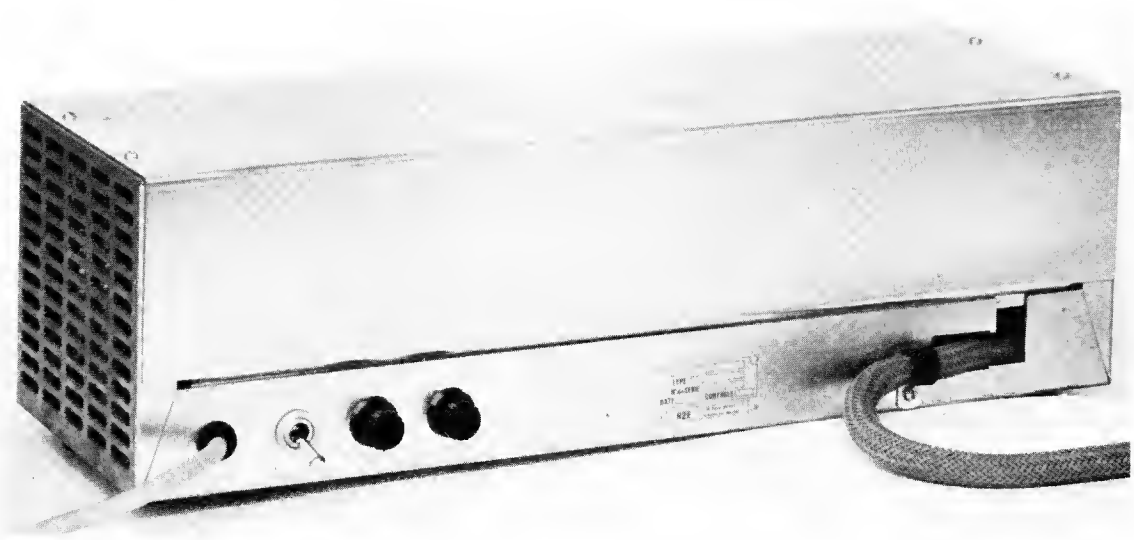


PHOTO 5 POWER SUPPLY

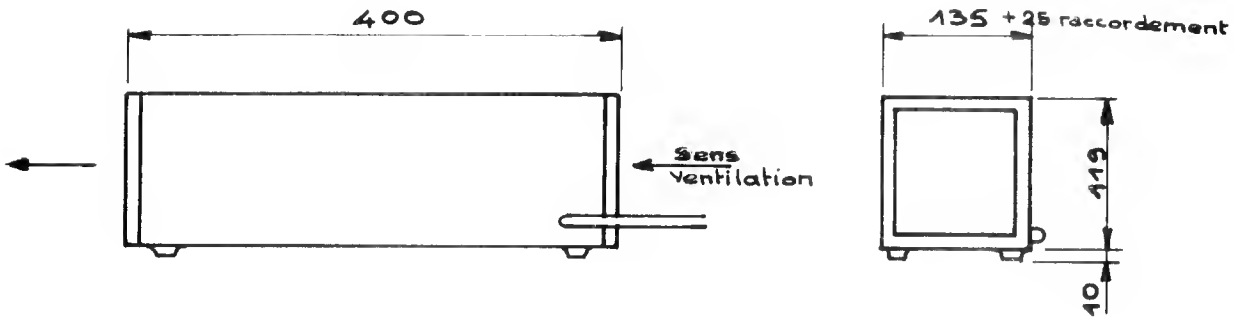
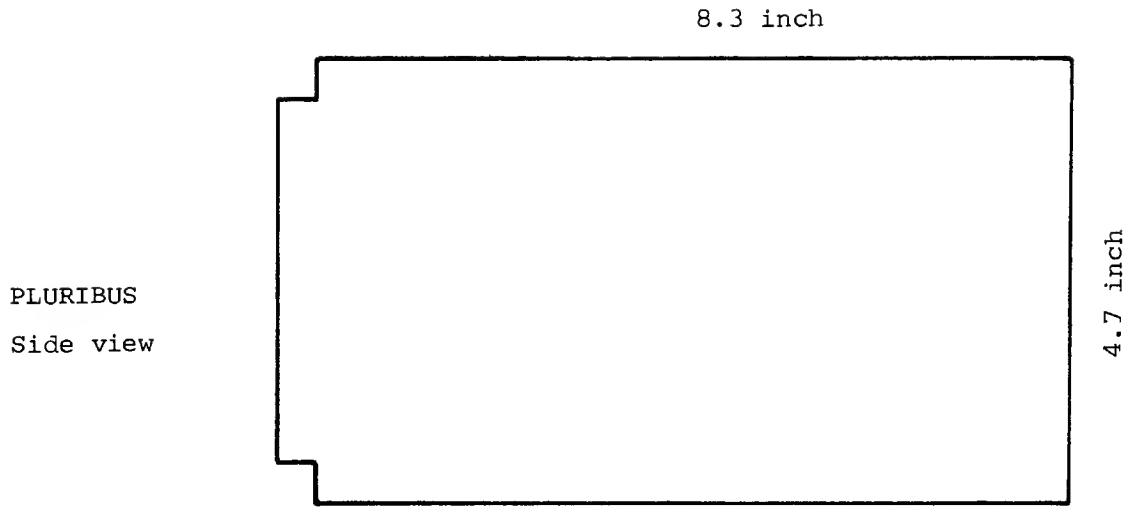
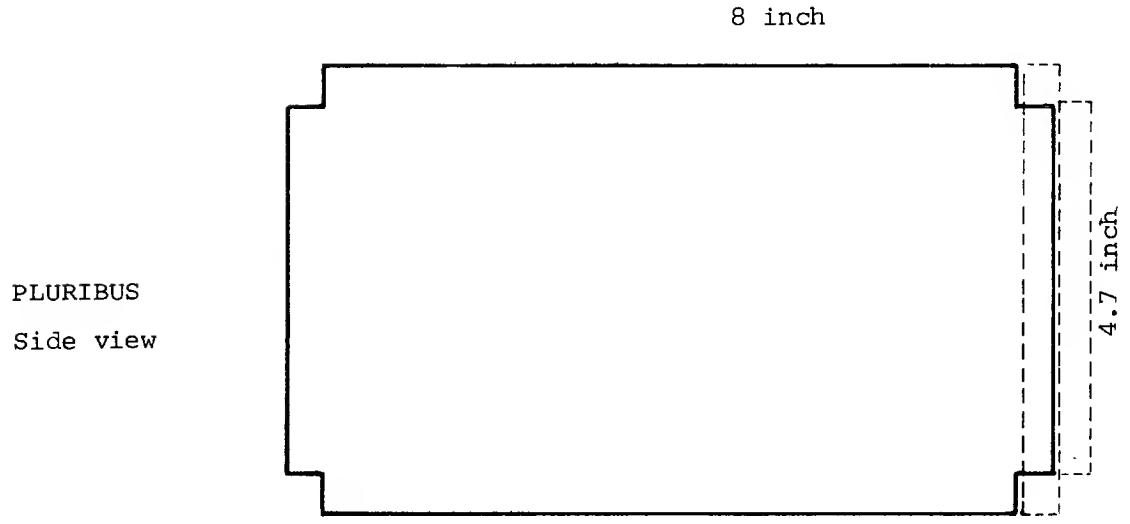


FIGURE 1.2 DIMENSIONS

Consumption : 250 W



Single connector card



Double connector card

Figure 1.4. MICRAL Standard Card

I.4. Construction

One the fundamental features of MICRAL is its ability to provide memory capacity and the number of interfaces strictly necessary to solve a particular problem. The construction of the computer is therefore highly modular.

Picture 1.4 shows one card and its dimensions.

For example, the Random Access Memory can be made of cards containing each a maximum of 2 K bytes. When more than 2 K is required, the system will have to be equipped with more than one RAM card. All RAMS being identical, the user will have to assign them an address so to enable the processor to address them correctly.

The following paragraph explains how the address assignment system works in MICRAL.

I.5. Address assignment

Generally speaking, each card has several cards (memories or interfaces). Decoders allow one to select the addressed card among the total of cards included on the card. A selector circuit allows one to select the card when the addressed card is included in that card.

The address of the card is taken from SO/ to S13/ pluribus Terminals, the low order bits address the card on the card and the high order bits the card in the system.

I.5.1. Memory addresses assignment

The fourteen bits of the address register allow one to address 16 K bytes as follows :

SO/TO S7/ address the byte on the page;
S8/to S10/ (in 2 K RAM card) or
S8/to S11/ (in 4 K RAM card) address the page in the card and
S11/to S13/ or S12/to S13/ address the card in the system.

The selector circuit works on the latter bits. On each bit number "i", it is possible to feed Si or Si/ depending on whether a "1" or "0" is to be assigned to this bit. This is done with an integrated circuit support on which are connected, on one side, facing each other, one input of the selector and circuit and, on the other, Si and Si/ signals. A jumper allows the user to connect the Si to the input when a "1" is to be assigned or the Si/ when a zero is to be assigned.

Example : place the jumper to assign a 2 K RAM card in a such way that page 1F is included. F will address the seventh page of the card and "1" will address the card in the system. The binary address being 001, S13/, S12/ and S11 should be connected to the selector inputs and the jumper placed as shown on figure 1-5.

I.5.2. Coupler cards

S9/ to S13/ address the output devices and SO/ to S7/ the input devices while S9/ to S11/ address the group of input devices.

The description of coupler address assignment is described in the corresponding paragraph.

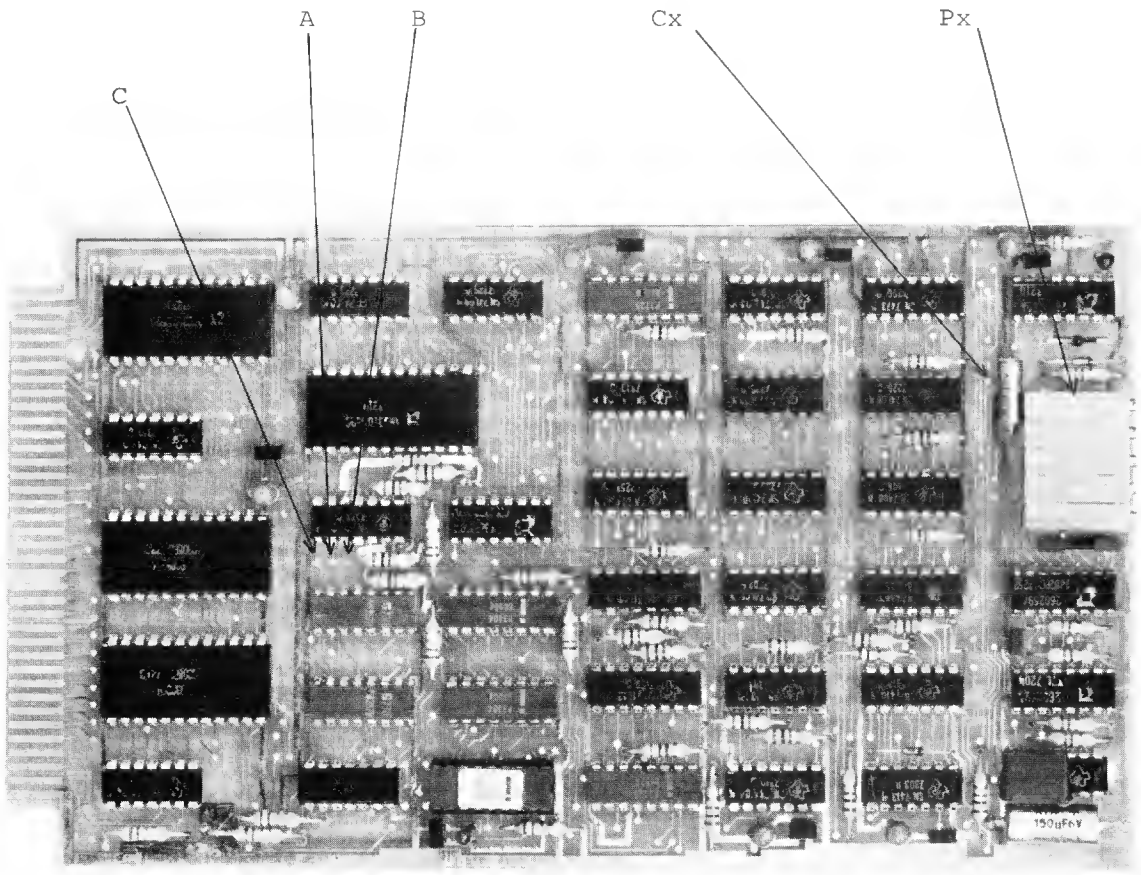


PHOTO 2.1 PROCESSOR CARD

II - PROCESSOR CARD M1100

II.1. Hardware description

Photograph 2.1. shows the processing card.
It has a micro-processor 8008 and its associated circuits allowing the decoding of the machine states and synchronization. Also included in the processor is the interrupt system, a real time clock, the automatic start-up circuits and the general address register of fourteen bits permitting the direct address of to 16 K memory words and the control of the I/O, as well as a control light diode.
Diagram 2.2 shows schematically the card's layout.

II.1.1. Registers

The processor contains seven registers addressable by the internal instructions :

A	the accumulator
B C D E	auxiliary registers
H L	auxiliary registers allowing to address the main memory.

These seven registers have an internal address which is respectively 0 - 1 - 2 - 3 - 4 - 5 - 6. Address 7 relates to the memory M addressed by registers H and L. H relates to the higher order bits of the address (six bits) and L to the lower order bits of the address (eight bits).

II.1.2. The program counter

The program counter is located in a push-down stack consisting of a Random Access Memory of eight registers of 14 bits addressed by a pointer. This pointer is decreased by each sub-routine call instruction (CAL or RST).

The new program counter (which is the address of the required sub-program) is recorded in the memory at the address determined by the pointer. The former program counter (i.e. the return address) is memorized in the address below, in the stack.

When a return instruction is decoded by the processor, the pointer is incremented. The program is then connected to the return address. This system permits the overlapping of up to seven sub-routines without losing the return address of the main program.

II.1.3. Condition Flags

Four flags are simultaneously set to each operation.

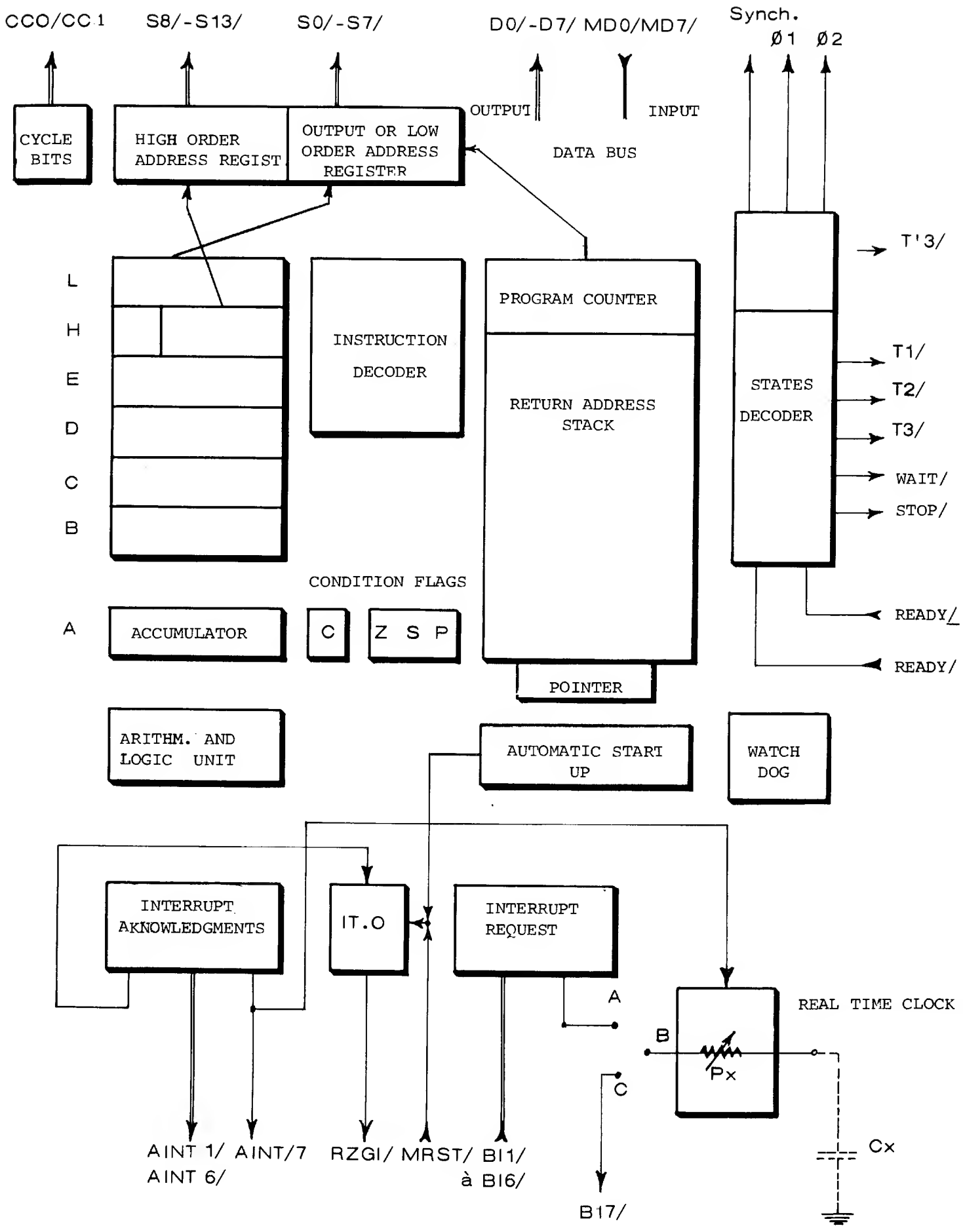


FIG. II.1 - PROCESSOR BLOCK DIAGRAM

Carry : this flag is set to "1" when there is an under or overflow on the accumulator.
It is set to zero by logical operations. It is not affected by load instructions or by increment or by decrement of the registers (B.C.D.E.H.L.).

Zero : this flag is set to "1" when the result of an operation is zero.

Sign : this flag is set to "1" when an operation leads to a result where the highest order bit is "1".
The term "sign" has been kept, although it is incorrect, analogically with the 2's complement code in which the highest order bit is the sign.

Parity: the flag is set to "1" when the result of an operation gives a result where the bits count is even.

In order to avoid any ambiguity, diagram 2.1. shows the state of flags C Z and S after a substraction or comparison instruction for the various cases : accumulator lower, equal or superior to the ground.

Picture 2.1. Position of flag after one operation SUr or CPr.

In the event the user wants the 2'S complement code, this diagram does not apply.

The conditional call or jump instructions allow these flags to be tested and to connect the program according to these states.

II.1.4. Interrupt system

Each interrupt level relates to one input, except level zero (called "catastrophic"), which is always active, each level can be enabled or disenabled individually, masked or unmasked collectively.

When an interrupt is activated, the program is connected to one of the eight adresses of page 0 :

/0 - /8 - /10 - /18 - /20 - /28 - /30 - /38

(Addresses are specified in hexadecimal when preceded by a/).

Each address is the beginning of a sub-program including up to eight sequencial instructions (to avoid overflowing into an higher level address). The first instruction of each of these sub-programs is not interruptible. Level 7 can be connected to the real time clock (connect A to B). The interrupt system includes also eight acknowledgement outputs.

II.1.5. The real time clock

The real time clock can be adjusted from 100 micro-seconds to one hour, with infinite variation. In the standard configuration, the internal clock is supplied with a 10 m.sec. period. The roughing out is done with the CX capacitor which must be adjusted to the desired duration scale. Fine adjustment is made with a PX potentiometer.

II.1.6. Synchronisation signals

II.1.6.1. PRINCIPLE

Knowledge of the synchronization system is not a must for the user. As a matter of fact, MICRAL operates under the following principle : at each addressing phase (for a fetch cycle, as well as for exchange of data between processor and memories, or processor and peripherals), the processor stops in a WAIT state, until the addressed device has sent a low voltage signal back to READY input of the pluribus which sets the processor back to active working, in synchronization with the time base. The WAIT state has a duration which is a multiple of the clock period. If READY comes early enough, the WAIT state is skipped. To obtain this result, the low voltage signal must be sent back to READY input at the latest 500 nanoseconds after the higher part of the address has been received from the processor on the pluribus.

When the processor is in the WAIT state, this is signified by a low voltage signal to the pluribus'WAIT/output.
The WAIT cycle is used to obtain a "step" through the console. The console sends back a low voltage signal to READY input which systematically holds the processor in a WAIT state until an high level voltage (5 V) has been sent back by means of the step by step key on the console (see console description).

II.1.6.2. The time base has two phases which are used as a reference. The processor returns a synchronisation voltage of half the SYNC period. SYNC/ period is the basic period for the machine (4 micro-seconds). The instruction is divided into memory cycles, themselves divided into machine states.

II.1.6.3. Memory cycles : there are four types of memory cycles (OP, READ, WRITE, I/O) which are defined by the state of the two CC0 and CC1 outputs as shown on diagram 2.2.

Fetch cycle

This cycle corresponds to the addressing of the memory during the fetch phase.

	A r	A = r	A r
C	1	0	0
Z	0	1	0
S	1	0	0

Figure 2.1 Position of flag after one operation SUr or CPr

Cycle	CC 0	CC 1
OP	0	0
LEC	0	1
ECR	1	1
E/S	1	0

Figure 2.2 Memory cycles coding

READ Cycle

This cycle relates to the addressing of the memory when reading data.

WRITE Cycle

It is in operation when the program requires writing of data into the memory.

I/O Cycle

This cycle refers to an exchange of data between the processor and an input or output. It includes an addressing phase of the peripheral and an exchange phase.

II.1.6.4. Machine states : cycles are composed of states, being multiples of the basic period.
Five states last for one basic period (T1, T2, T3, T4, T5) and two states have a duration depending on the system. One of them is the WAIT state mentioned above, the other is the STOP state.

- States of fixed duration :

  . T1 Addressing of the lower order part of the address.
  . T2 Addressing of its higher order part.
  . T3 Instruction decoding.
  . T4 Instruction execution.
  . T5 " "

- States of indefinite duration :

  . WAIT state described above.
  . STOP : the computer is in an idle state necessitating re-starting by means of an interrupt.

II.1.7. Automatic start

The automatic start system consists of detecting the raise of the power supply voltages and generating a zero level interrupt (INIT), which connects the program back to zero address. A restart is also obtained by generating zero level interrupt. The same result can be obtained by depressing the INIT key from the console.

II.1.8. Optical "watch-dog" signal

As just indicated, zero level interrupt cannot be disenabled. Now, bit 0 of OUTPUT /17 which is not used to activate the interrupt level zero, controls a flag connected to a light emitting diode which acts as an optical "watch-dog" signal. State "1" of the flag relates to the light being extinguished.

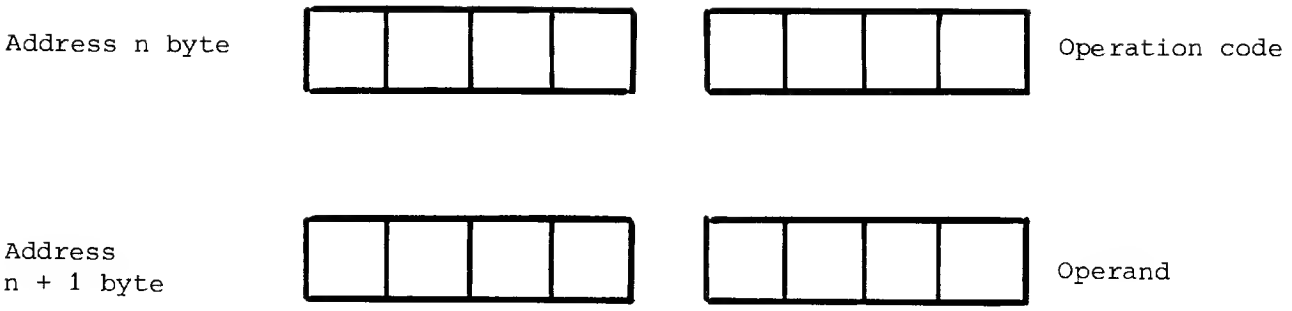


Figure 2.3 TWO BYTES INSTRUCTION STRUCTURE

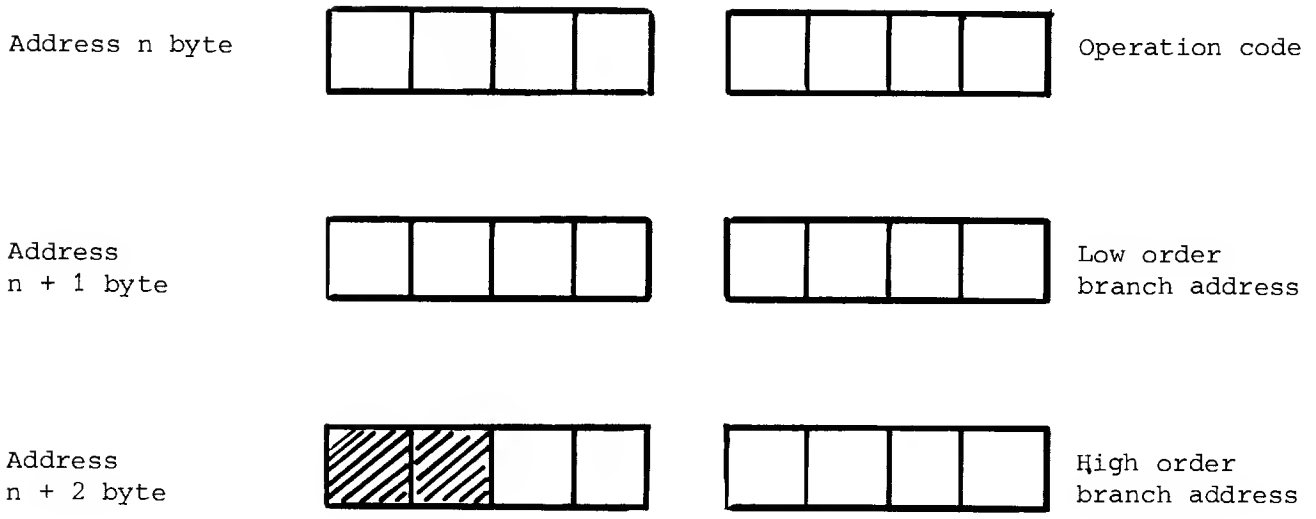


Figure 2.4. THREE BYTES INSTRUCTION STRUCTURE

Incidentally, this device is used for the 1, 2, 3, 4 maintenance program of the MICRAL library. In order to check MICRAL functionnally, load this program and ensure that the diode lights up according to program.

II.2. Description of instructions

II.2.1. Structure of instructions

MICRAL instructions are generally contained in a byte. Some instructions are contained in two bytes. These instructions use as an argument data located in the byte immediately following the operation code (see diagram 2.3).

On the other hand (see diagram 2.4) connecting instructions (jumps or calls) are contained in three bytes, the first containing the operation code, the second the lower order address (eight bits), the third the page address (six bits, the two heaviest bits being inoperative).

Instructions can be divided into several groups:

- loadings.
- arithmetical and logical operations.
- connections.
- input output.
- machine instructions.

These are also pseudo-instructions of the assembler that will be examined in chapter relating to the software.

In the following paragraphs, instructions are described group by group.

II.2.2. Loading instructions

II.2.2.1. General structure : in these instructions, each register is designated by its internal code (A=0, B=1, C=2, D=3, E=4, H=5, L=6). In memory reference instructions, the address shown by H and L, called M, is designated by code 7. Simple loading instructions are contained in one byte, while immediate loading instructions are contained in two bytes.

II.2.2.2. Simple loading instructions :

The commonest form of simple loading is :

$$L\ r_1\ r_2$$

It means load r_1 with the content of r_2 (the latter remaining unchanged with :

r_1 = A, B, C, D, E, H, L, M

r_2 = A B C D E H L M

As can be seen, this instruction includes two operation code bits, (the heaviest ones), and two zones of three bits which are internal addresses of the micro-processor (see diagrams 2.5 and 2.6).

Except LAA which is called NOP, Lrr type operations cannot be used as they perform operations such as masking and unmasking (see paragraph "machine instructions").

II.2.2.3. Immediate loading instructions :

The most general form of immediate loading instructions is :

L r I ARG

r = A B C D E H L or M

It means load register r with the content of argument ARG placed in the byte immediately following the instruction (see diagrams 2.7 and 2.8).

Diagram 2.8 demonstrates how the accumulator can be cleared by means of instruction LAI 0. We shall see later that another shorter instruction exists (XRA) enabling to clear the accumulator but with the inconvenience of modifying the flags.

II.2.3. Arithmetical and logical operations

These affect the flag on all operations, they divide into three categories :

a) increment or decrement of auxiliary registers;

b) arithmetical and logical operations on the accumulator;

c) rotations of the accumulator.

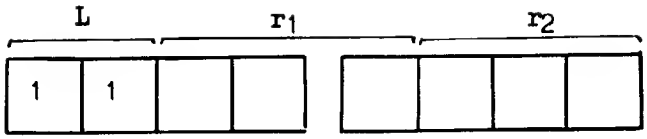
II.2.3.1. Increment decrement (diagrams 2.9 and 2.10) = the most general form of increment decrement instructions, is as follows :

IN r

DC r

r = B C D E H or L

They mean : increment (IN) or decrement (DC) the register r by one, and set Z,S and P flag according to the result, C remaining unchanged.



	R R R	
A	0 0 0	$R_1R_1R_1$ Register
B	0 0 1	$R_2R_2R_2$ Register
C	0 1 0	
D	0 1 1	
E	1 0 0	
H	1 0 1	
L	1 1 0	
M	1 1 1	

Figure 2.5 - SIMPLE LOADING INSTRUCTION STRUCTURE

LAB	LOAD "A" WITH CONTENTS OF B
LCM	CHARGE "C" WITH MEMORY INDICATED BY H + L
LME	WRITE IN THE MEMORY INDICATED BY H AND L THE CONTENTS OF E

Figure 2.6 - STRAIGHT FORWARD LOADING EXAMPLES

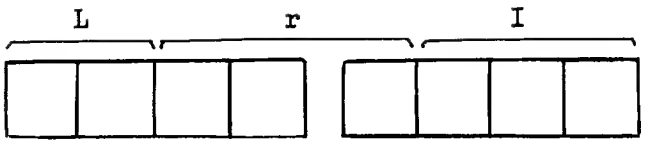


Figure 2.7 - IMMEDIATE LOADING INSTRUCTION STRUCTURE

```
LAI 0           RESET THE ACCUMULATOR AT ZERO
```

```
LMI 0           RESET THE MEMORY INDICATED BY H AND L
LLI1
```

```
LMI 0           RESET THE MEMORY INDICATED BY H AND L
LLI ADRB        LOAD THE LOW ADDRESS MEMORY ADRB
LHI ADRH        AND HIGH ADDRESS ADRH
```

Figure 2.8 - EXAMPLES OF IMMEDIATE LOADING

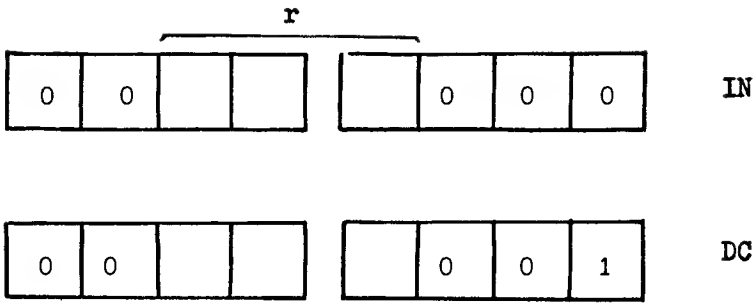
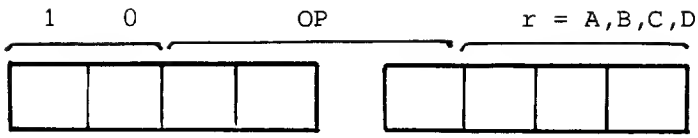


Figure 2.9 - INCREMENT AND DECREMENT INSTRUCTION STRUCTURE

```
LLI  0        FOLLOWING THESE INSTRUCTIONS L CONTAINS 1,
INL           CARRY IS UNAFFECTED Z, S AND P ARE FALSE
```

Figure 2.10 - INCREMENT INSTRUCTION EXAMPLES



OP :	AD	0	0	0	ADD A and r
	AC	0	0	1	ADD A, r and C (Carry)
	SU	0	1	0	Subtract r from A
	SB	0	1	1	Subtract r + C (Carry) from A
	* ND	1	0	0	And Logic between A and r
	* XR	1	0	1	Or exclusive between A and r
	* OR	1	1	0	Or inclusive between A and r
	CP	1	1	1	Compare : Set the flags in accordance of the difference A - r without modifiing accumulator.

Figure 2.11 STRAIGHT FORWARD OPERATION INSTRUCTION STRUCTURE

```
LAM                 THIS PROGRAM ALLOWS TWO NUMBERS TO BE
ADB                 ADDED IN DOUBLE PRECISION ONE LAYS BETWEEN
LMA                 TWO CONSECUTIVE MEMORY CASES THE
INL                 OTHER BETWEEN THE REGISTER B AND C
LAM
ACC
LMA
```

Figure 2.12 EXAMPLE SHOWING AN OPERATION

It is important to note that these instructions do not allow the modification of A or M.

II.2.3.2. Simple arithmetical operations (diagrams 2.11 and 2.12) : these instructions use two operands. One is contained in the accumulator, the other is designated by the instruction of either one of the registers A or M. The result is stored in A and all the controls are amended as a result.

To demonstrate the instruction form, we use an auxiliary symbol OP (not recognized by the assembler) as representing one of the eight operation codes, accepted by the processor.

The OPr form means : perform operation OP between A and r, the result is stored in A.

The eight of codes are :

- AD for addition A and r;
- AC for addition A, r and C (Carry);
- SU for substraction r from A;
- SB for substraction r + C (Carry) from A;
- ND for and LOGICAL BETWEEN A and r;
- XR for or exclusive between A and r;
- OR for INCLUSIVE BETWEEN A and r;
- CP compare A and r and set the controls by the result of A-r without modifying A or r.

Note that in the example shown in diagram 2.12, the use of INL does not affect the condition of the carry flag thereby permitting the transfer of the instruction AC. In these operations, all flags are set by the operation's result. Where logical operations are concerned (as marked with an asterisk), the carry control is reset to zero.

II.2.3.3. Immediate operations : the usual form, OPI ARG, means : perform operation OP between the content of the accumulator and the argument contained in the memory byte immediately following the operation code of the instruction. The controls are set as per previous paragraph (see diagrams 2.13 and 2.14).

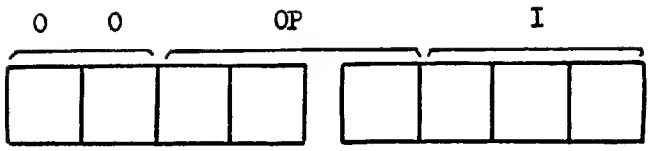


Figure 2.13 - FIRST BYTE OF AN IMMEDIATE OPERATION STRUCTURE

```
NDI OF          MASKS THE FOUR HIGHEST ORDER BITS
ADA             CLEARS ACCUMULATOR AND SETS FLAGS ON 0
XRA
NDA             SETS FLAGS IN ACCORDANCE WITH CONTENT OF A.

CPA
```

Figure 2.14 - EXAMPLES OF OPERATIONS

II.2.3.4. Accumulator's rotations : the rotation operations of the accumulator are described in diagram 2.15.

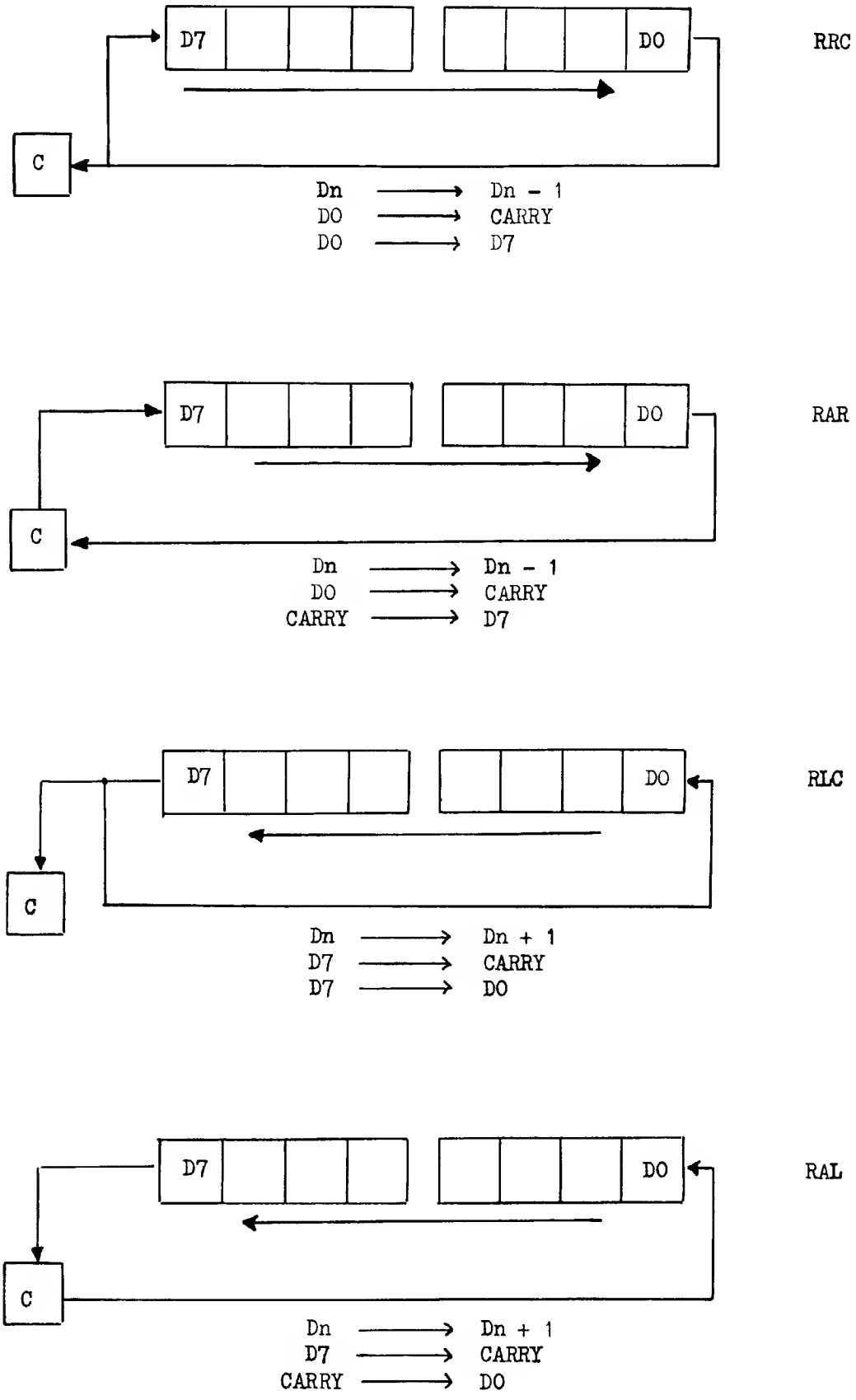
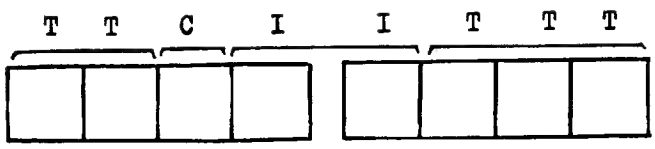


Figure 2.15 - ROTATIONS



T = TYPE	0	1				0	0	Conditionnal
	0	1				1	0	"
	0	0				1	1	"
	0	1				0	0	Unconditionnal
	0	1				1	0	"
	0	0				1	1	"
C = CONDITION			0					FALSE
			1					TRUE
I = INDICATEUR FLAG			0	0				CARRY
			0	1				ZERO
			1	0				SIGN
			1	1				PARITY

WHEN CONNECTED, UNCONDITIONALLY THE CII BITS ARE THE SAME EXCEPT FOR RET FOR WHICH THE 011 CONFIGURATION CANNOT BE USED IT GIVES REI.

Figure 2.16 - BRANCH INSTRUCTION STRUCTURE

JFC ARR	JUMP TO ADDRESS IF THE CARRY IS IN A FALSE POSITION ("0" LOGIC)
RTZ	RETURN TO THE MAIN PROGRAMM IF THE ZERO ES IN THE TRUE POSITION

Figure 2.17 - EXAMPLES OF CONDITIONAL CONNECTIONS

B

RST 4 CALLS SUBROUTINE LOCATED AT ADDRESS /18

Figure 2.18 - RST INSTRUCTION EXAMPLE

Figure 2.19 - RST INSTRUCTION STRUCTURE

II.2.4. Connection instructions

II.2.4.1. Unconditional connections : there are three types of unconditional connections :

JMP - CAL - RET

JMP connects the program unconditionally to the address contained in the two bytes immediately following the OP code. The first byte contains the eight bits of the lower order address and the second byte the six bits of the page address.
The code for the first byte is OIXXXIOO (X being either 1 or 0).

CAL has the same effect, but it memorizes the address of the program counter in the micro-processor's stack. This instruction allows calling of a sub-program. The code for the first byte is OIXXXIIO.

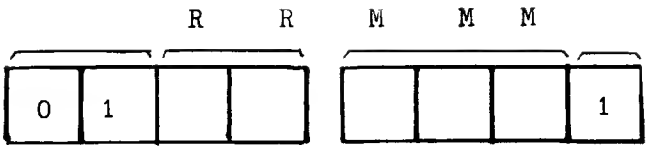
RET connects the program to the first address memorized in the stack. This instruction is used to return from a sub-program. The code is OOXXXIII (X being immaterial except configuration OII).

II.2.4.2. Conditional connections : there are three instructions in this category.

The first letter of the mnemonic instruction defines the type of operation (J for JMP, C for CAL, R for RET), the second letter defines the condition, T (True) or F (False) depending on whether the connection is desired upon the true or false condition, and the third letter specifies which control is to be tested to make the connection. Diagram 2.16 shows the structure of the connection instructions and diagram 2.17 two examples of conditional connections.

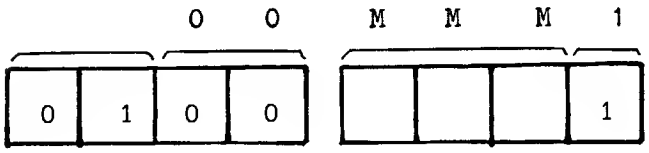
II.2.4.3. RST (CAL included in one byte) : there is another unconditional connection instruction which permits, in one single byte, the program to connect with one of eight addresses on the zero page.

This instruction is written : RST N, N being an argument ranging from 0 to 7, the program is connected to address 8 N.
See example (diagram 2.18) and structure (diagram 2.19). This instruction in spite of its form, is contained in one single byte. The argument is inserted into the instruction's code.



RRMMM is the device address, RR must be different to 00

Figure 2.20 - OUTPUT INSTRUCTION STRUCTURE



MMM is the group device ADDR

Figure 2.21 - INPUT INSTRUCTION STRUCTURE

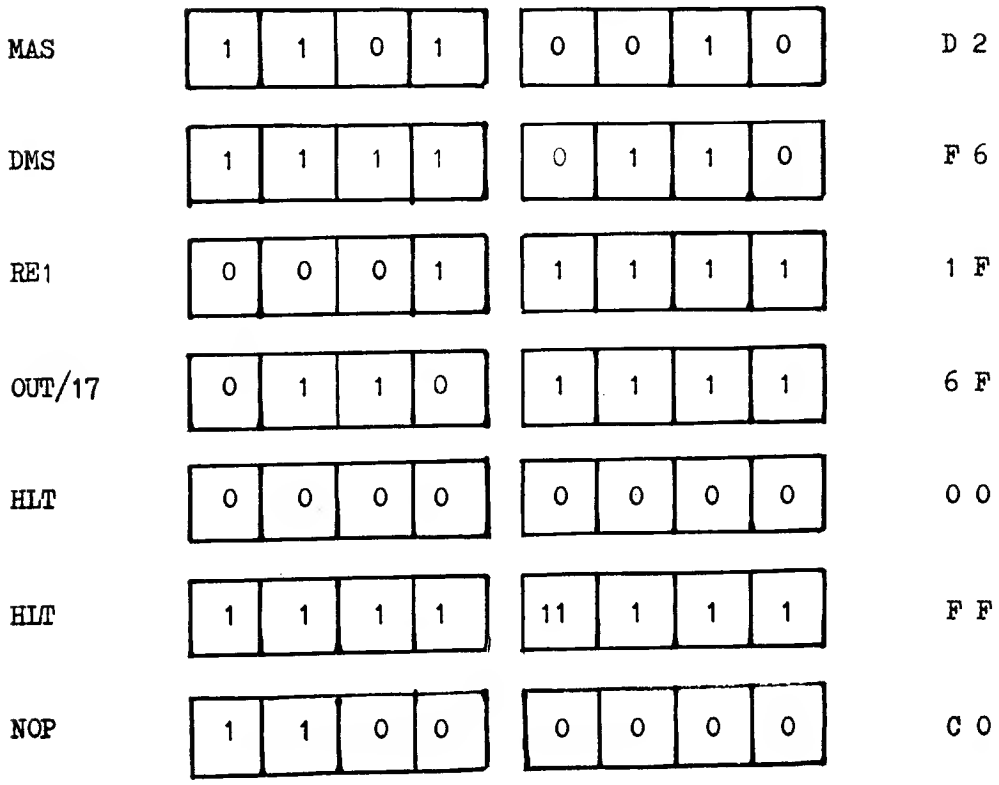


Figure 2.22 - MACHINE INSTRUCTION CODES

II.2.5. Input/output instructions

II.2.5.1. Outputs : there are 23 output instructions. They are in the OUT DEV form.
DEV is an argument inserted into the operation code representing the address of a card and necessarily ranging from 0 to 22 decimals (equivalent to 0 and /17 hexadecimal).
The 23rd address is used to enable interrupt (paragraph II.26.4.).
Instruction RST (and instruction INP described below), also instruction OUT, in spite of its form, are located in one single byte of memory. Five bits are reserved for the argument, within the operation code.

II.2.5.2. Inputs : except when addressing a channel (see paragraph V), the input operation takes place in two steps. First, A, is precharged by an immediate loading instruction, then, instruction INP GRP is utilized.
GRP is an argument inserted into the operation code representing the address of a group of inputs, ranging from 0 to 7. Each group includes 255 byte inputs, addressed by precharging A. It is possible to address 7 x 256, i.e. 1792 input bytes, equal to 14.336 single wire inputs.

Example :

```
LAI 12
INP 12
```

gives input 12 of group 2.

Channels are called by their group address and thereby neutralize the 256 inputs corresponding to their group address.

II.2.6. Machine instructions

II.2.6.1. Instructions MAS : instruction MAS allows masking of the interrupts, i.e. it disengages all interrupt levels, except level 0.

II.2.6.2. Instructions DMS : instruction DMS activates all the enabled interrupt levels, making them all operational. Immediately after instruction DMS has been executed, the program can be interrupted by the level of interrupt that holds the highest priority.

II.2.6.3. Return instruction REI : this instruction permits the performance of two operations simultaneously : one unconditional RET and one DMS.
It avoids losing a level interrupt in certain instances (a regular interruption between DMS and RET without REI).

II.2.6.4. Enabling/Disenabling :

This instruction is an output instruction on address /17. It performs two functions :

1°) applying a signal to the optical "watch-dog" through bit 0;

2°) arming the interrupt levels in accordance with the configuration contained in the accumulator.

A level of priority corresponds to each bit :

Bit	0	Watch-dog
Bit	1	Level 1
Bit	7	Level 7

one "0" disenables the corresponding level.

One "1" re-enables it.

Example : LA1 04

OUT/17

Applies 0 to the watch-dog and disarms all levels except level 2.

LA1 0F applies A to the watch-dog and activates levels 1, 2 and 3.

II.2.6.5. Instructions HLT : renders the processor into a STOP condition.

Instruction NOP : it is inoperative.

PROCESSOR CARD

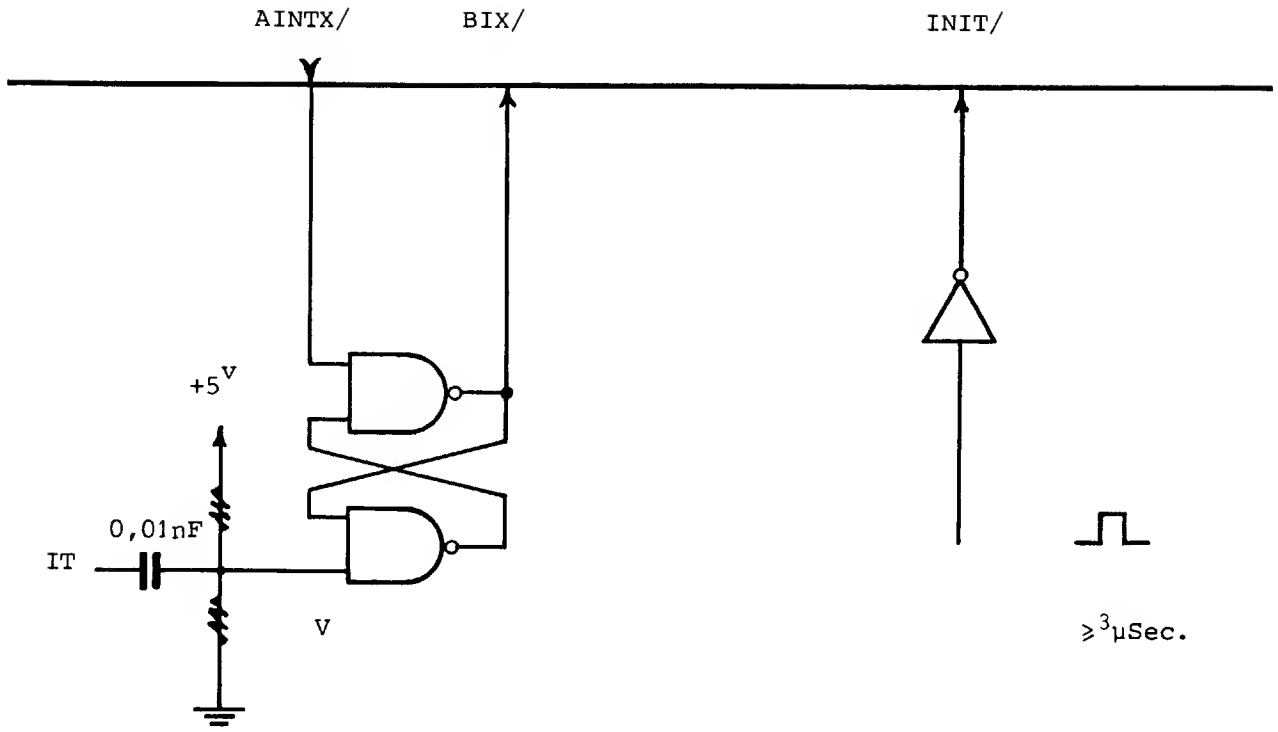


Figure 2.23 OUTSIDE PROCESSOR CIRCUITS

II.3. Functioning of the Processor Board

II.3.1. Introduction

The general system, based on the request/acknowledgement principle, allows the user to ignore the functioning of the machine states.

When an interface card is addressed, it sends a low voltage signal to the READY/Terminal indicating that it is ready to supply or to collect the requested information.

If this signal does not arrive before T'3, the processor enters a wait state. Following the circumstances, the supplied information is accepted by the processor in the T'3 state which follows immediately upon addressing or after one or several WAIT cycles.

The processor then executes the instruction either in the current cycle, either in a new memory cycle.

A more detailed description is provided in the appendix 2 and 3 (example related to CFC instruction).

II.3.2. Interrupts :

II.3.2.1. Principle : MICRAL interrupt system is graded in eight levels, Level zero, designated by INIT, called catastrophic, cannot be masked nor disenabled. The others can be masked and unmasked as a whole, and individually enabled or disenabled by meansof an internal mask generated by an output instruction whose address is /17 (23 decimal).

Interrupt must be used as follows (diagram 2.2.3.) :

- a Flag, outside the processor card, must memorize the interrupt request corresponding to each level and be reset by the corresponding recognition signal.

Reset takes place as soon as the interrupt is recognized.

II.3.2.2. Priorities :

The hierarchy of priorities at the processor's level is effective only at recognition and not at execution. It means that in case several interrupt levels show up together or are present together after unmasking, the one that will be recognized is the one holding the highest priority, in the order 7, 6, 5, 4, 3, 2, 1, 0.

If the first instruction of the interrupt program (which cannot be interrupted) is not a general masking "MAS", the program will be instantly interrupted by any of the recorder interrupt levels, even if their priority is lower to the one currently being executed.

If the first instruction is a mask (MAS, it is then possible to proceed with a dynamic servicing of the interrupt priority through "individual enable" instruction (OUT/17).

II.3.2.3. Execution :

When an interrupt is requested, and assuming it is enabled and unmasked, the processor will recognize it.

If it is disenabled, it will be recognized only after arming has taken place and in the absence of masking. If masking takes place, it will be recognized when unmasking takes place if there is no higher priority level request.

When an interruption has been carried out, the current instruction is executed even if it includes several cycles. Afterwards, the program is connected to an address of page O which depends on the level as shown on diagram below :

Niveau	Adresse D	Adresse H
0	0	/00
1	8	/08
2	16	/10
3	24	/18
4	32	/20
5	40	/28
6	48	/30
7	56	/38

```
        ORG/18
        MAS             INTERRUPT MASKING
        JMP SV3         JUMP TO SV3

SV3     XXX             SAVEGARD PROGRAM
        XXX

        XXX
        DMS             INTERRUPT UN MASKING
        XXX             INTERRUPT PROGRAM

        XXX
        MAS             INTERRUPT MASKING
RS3     XXX             RESTITUTION PROGRAM
        XXX

        XXX
        REI             RETURN WITH UN MASKING
```

Figure 2.24

Examples of interrupt program :

Let's assume that level 3 interrupt program is to be written.

When an interrupt takes place, the program will be connected to instruction /18 (24 decimal).

Diagram 2.24 shows the program which will have to be used. In this program, interrupt have been masked during the safe-guarding and program restitution (respectively SV3 and RS3). During the interrupt program, interruption mask has not been modified and the program can be interrupted by all the enabled interrupt levels, whatever their priority may be.

Note that a jump instruction is used almost immediately to avoid overlapping into the level 4 interrupt program starting at address /20.

DIAGRAM 2.25

```
        ORG    24
        MAS
        LAI  / 02
        OUT    23
        DMS
        JMP    IT3

IT 3    ---
        --
        --
```

In that example, prior to entering the interrupt program itself, interrupt level 2 is disenabled and the "watch-dog" is switched-on.
As soon as DMS instruction is executed, the program can be interrupted by the interrupt levels 2 and 0 exclusively.

Note that the program starting at address /18 precisely contains 8 instructions.

Two more complete interrupt examples will be covered in the chapter related to the use of the channel stack.

II.3.3. Use of the real-time clock :

Real-time clock yields impulsions at a frequency determined by the time constant composed of a capacitor and infinite resolution multi-turn potentiometer.

Diagram 2.26 indicates the value of the capacitor to be used to obtain a desired range of frequency. The temperature stability of the capacitor is in direct relation with the clock's.

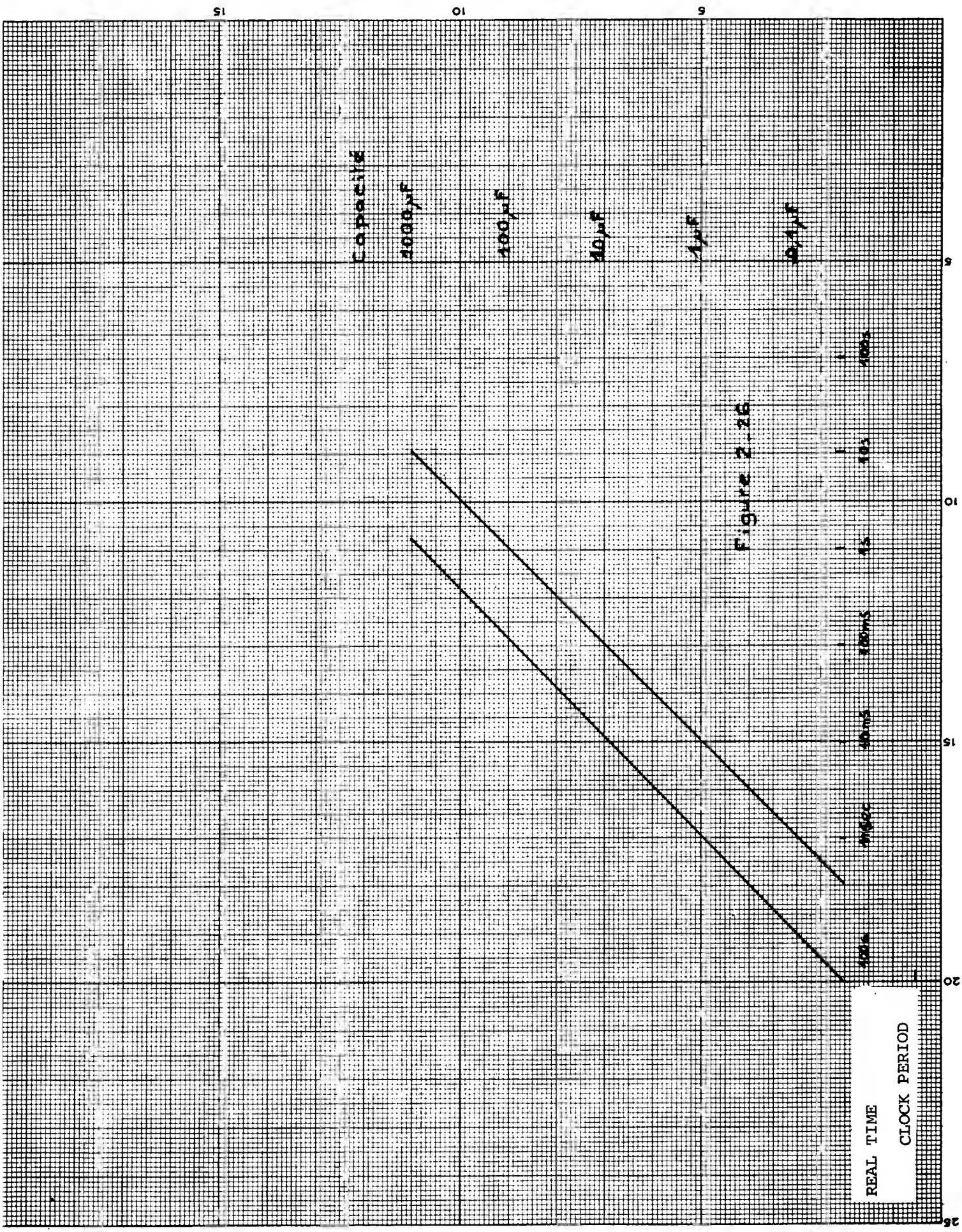

Capacitor
2000μF
200μF
10μF
1μF
0.1μF
1000s
100s
10s
1s
100ms
10ms
1ms
100μs
REAL TIME
CLOCK PERIOD

Figure 2-26

The potentiometer allows fine adjustment of the frequency within a 10/1 ratio.

The scale capacitor and the adjustment potentiometer are arrowed on stet. The four other potentiometers must remain untouched by the user. They are used for time base adjustment at the factory.

In the standard version, the internal clock is supplied at 10 milliseconds. Real-time clock can be connected on level 7 interrupt. This is obtained by wiring points A and B of the processor card (see photo 2.1). The processor card is supplied with this connection wired (except when MICRAL is delivered in a system where real time clock must not be used on interrupt). It is not necessary to connect the recognition output, as it is connected automatically on the card.

To disconnect the clock, simply disconnect A and B. It is important to note that the clock is triggable. It means that the first pulse following unmasking or enabling of level 7 is positioned at its nominal value.

II.4. Technical specifications of the processing card

II.4.1. Consumptions :

. Typical values :

5 V : 650 mA

- 9 V : 27 mA

II.4.2. Maximum loads on outputs :

. Group 1 : A INT 0/ to A INT/ 1/

. Group 2 : I_1 / I_2 / SYNC

T_1 / T_2 / T_3 / T'_3 / WAIT STOP

RGZ 1/ - Clock output /

SO/ to SO_{13} / / DO/ to D_2 /

Group 1 outputs have a single 74 series TTL input.

Group 2 outputs have a 74L series input for each connector, equal to a maximum of 22 TTL 74L inputs.

Output specifications :

. Group 3 : BI/1 to B17/

. Group 4 : READY/ READY C/

INIT/

MDO/ to MD7/

Group 3 inputs are similar to a 74 series TTL input.

Group 4 inputs have a resistance of 470 Ohm connected to 5 V.

As a consequence, they accept 74 series TTL circuits with anopen collector.

Delay specifications :

READY : to avoid the processor entering a wait state, READY signal must return to zero at the latest 500 nanoseconds after the higher address has been delivered by the processor.

DATA : data must be given during T'3 state.

III - Memory cards

III.1. Memory description :

MICRAL's memory is divided into 64 pages of 256 bytes. Its construction is based on a modularity which allows increases page by page for the mixed RAM memory and 4 by 4 pages.

To achieve that, memories are composed of integrated circuits laid out on the memory cards through supports. Each card contains a selecter circuit enabling the processor to identify and choose it. The address is delivered through SO/ to S13/ stet, card address through S11/ to S13/, page within the card through S8/ to S10/ and Byte within the page through SO/ to S7/.

The data is received through DO/ to D7/ and supplied to MDO/ to MD7. Terminals CCO and CC1 indicate the type of cycle requested by the processor.
CCO = 0 indicates a read cycle - CC1 = 0 Fetch;
CC1 = data Fetch.

CCO = 1	CC1 = 1	indicates a write cycle,
CCO = 1	CCO = 0	indicates that the memory is not addressed.

As far as timings are concerned, the eight lower order bits address are delivered to the pluribus at T, CCO and CC1 and the six higher order bits address at T_2. Data is exchanged at T'_3.

The memory cards can be supplied blank or filled. It is in the user's interest to order fitted cards, because they will be fully tested in accordance with MICRAL's specifications.

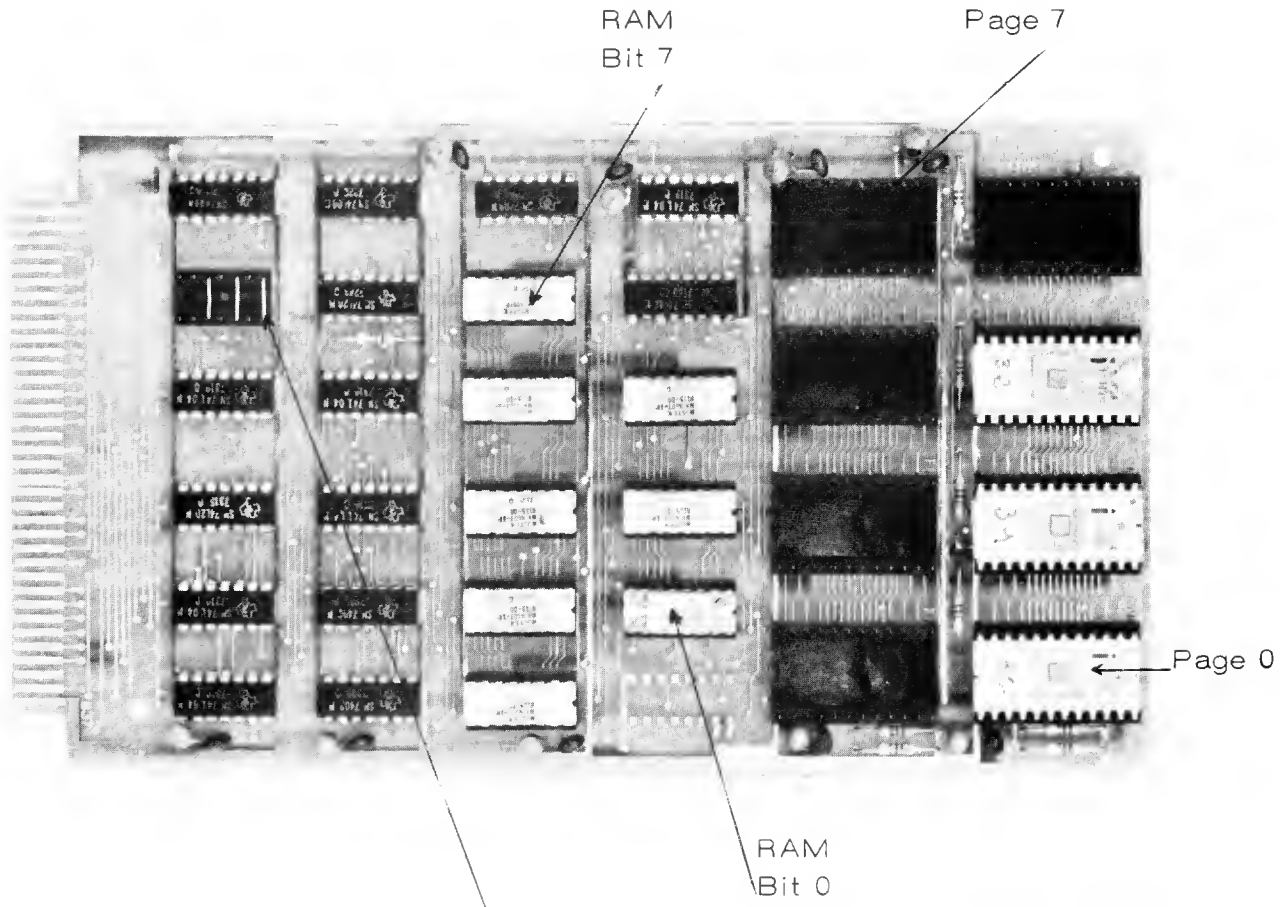
There are two types of memory cards namely :

. the mixed memory cards;

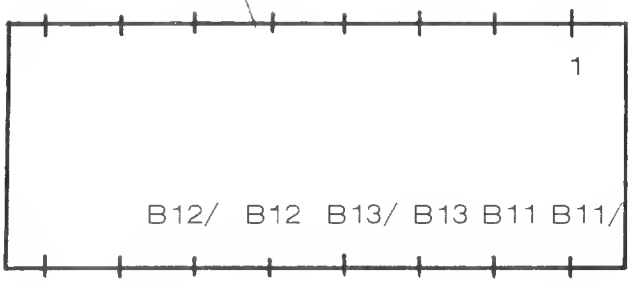
. the RAM memory cards.

III.2. MI200 mixed memory cards

The mixed memory card, like the RAM, can contain up to 8 pages. The first seven pages are Read only memories (ROM). The eighth page can be either a Random access page (RAM) or a ROM page. This is done by inserting the desired circuits in the proper supports. The ROM circuits must be 1302, 1602 A or 1702 A circuits, respectively,



MIXED MEMORY



ASSIGNED FUNCTIONS

ROM 7 or 8 pages

RAM 1 or 0 page

Total number of devices assigned : 8 pages or 2 K Bytes

* Page seven may be ram or rom depending on circuits incering into the supports

Figure 3.1 - Mixed memory address assignation

depending upon whether a ROM is desired (see paragraph 1-2 of introduction), a PROM or a REPROM. These circuits are organized with 256 bytes which therefore constitute the minimal ROM increment.

The RAM page is composed of eight 1101 or 4007 circuits. These circuits are, 256, one bit, Static memories. One circuit assigns the eight pages addresses; the latest page of the card can be either a RAM or a ROM. Photograph 3.1 shows the mixed memory card. Diagram 3.1 defines the assignment of the memories (see complete clarification in paragraph 1.5).

III.3. M1202 Random memory card (RAM)

The RAM memory card can contain up to 2048 bytes, made of 1024, one bit, static memories. The complete card represents 8 memory pages.

The assignment of the card address is achieved by a selecter circuit (photo 3.2.) which must be wired as indicated on diagram 3.2.
Diagram 3.2. shows the memory card and the geographical location of the words within that page (see complete clarification in paragraph 1.5.).

A voltage protecter option is provided on MICRAL. It is made of a battery coming in through terminal "Protected supply" (A 5 on the pluribus). When the + 5 V disappears, the protected supply voltage maintains a level enabling the information to be held within the RAM. As soon as the + 5 V reappears, the data can be reutilized. While the 5 V supply disappears, it is not possible to read nor to write into the memory.

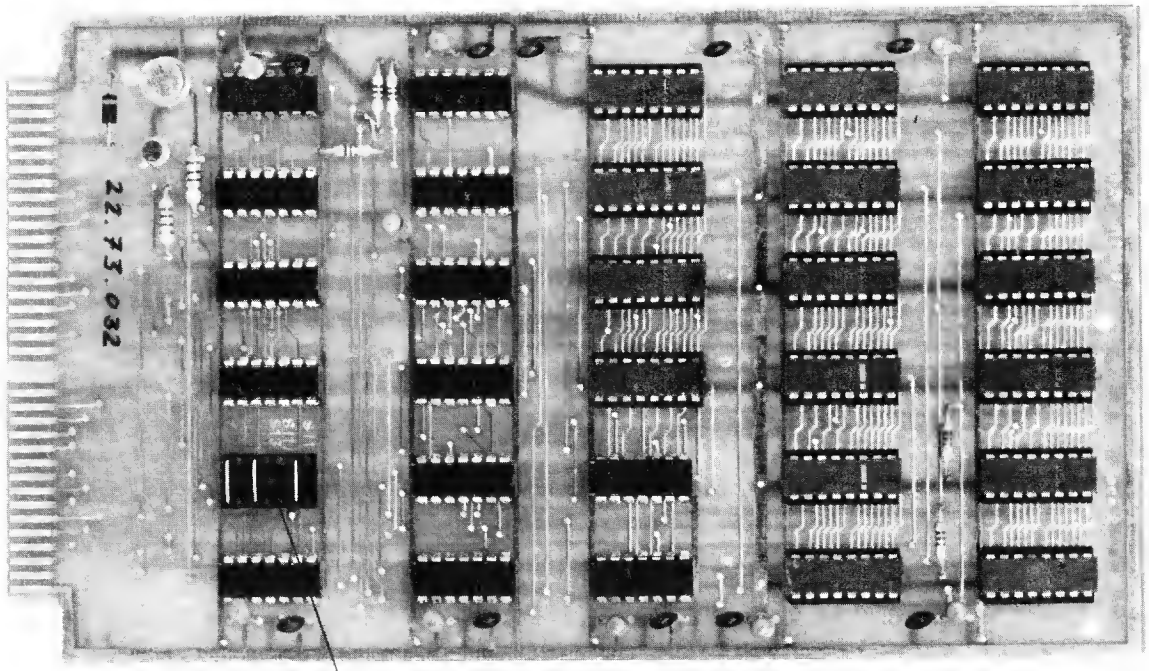
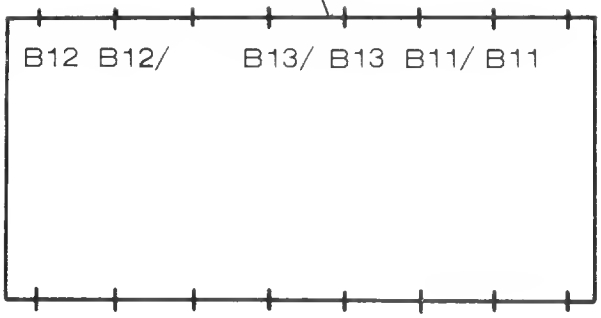


PHOTO 3.2



2K BYTES RAM CARD

ASSIGNED DEVICES RAM

NUMBER OF DEVICES ASSIGNED : 8 pages or 2K Bytes

Figure 3.2 2K RAM CARD ADDRESS ASSIGNATION

IV - Coupling cards

IV.1. Direct interfaces - Specialized interfaces

MICRAL's outside coupling is done through input output interfaces or through the stock channel.

Two types of interfaces are provided :

. Direct interfaces connected to the pluribus;

. Specialized interfaces connected to direct interfaces.

IV.1.1. Direct interfaces

All the direct interfaces are connected in negative logic. It means that a one in the accumulator, when transferred into an output, will yield a low level. Likewise, when in an input, a low level will yield one in the accumulator.

Regarding the instructions, each input and each output will generate a signal. This signal indicates that the input data has been taken over or that the output data has been memorized. The direct interfaces are controlled by 8, i.e. by channel, which will be called byte input or output, having 8 wires. The expressions input and output are reserved to a "one wire" input or output.

Electrically, the interfaces are connected as follows : with the MICRAL to the pluribus (terminals DO/ to DO7/ for outputs and MDO/ to MD7/ for inputs), and with the peripheral by means of a connector which is supplied with the card.

IV.1.2. Specialized interfaces

They are connected in negative logic to the calculator. With the peripherals, they are obviously adapted to the specified levels. Generally speaking, they pick the power supply exclusively from the pluribus. The input and output signals are connected at the end of the direct interface cards.

IV.2. Inputs and outputs capacity

In MICRAL, 8 groups of input and 24 output bytes can be addressed directly. One of them, the 23, is used for enabling the interrupt levels. Each input and each output include eight wires. Besides, in MICRAL subaddressing of the inputs is obtained by precharging the accumulator. This allows addressing of up to 7 times 256 inputs of 8 wires.(At least, one group address is supposed to be used for a stack channel). Various interface cards allow these channels to be used.

IV.2.1. Addressing of inputs/outputs

At input, the group's address is on bits S9/, S10/ and S11/ of the pluribus, the channels address on bits S0/ to S7/ (S12/ and S13/ are of necessity 00).

At output, the address is on bits S9/, S10/, S11/, S12/ and S13/.
The pattern S12 = 0, S13 = 0 is not used, being reserved for inputs.

Each interface has one or several selection circuits to assign the address of its channels.

The principle of interface assignment is as follows :

- for inputs, circuits are used to assign group and channel. This is done with jumpers on a support as with memories. For ease of identification, we shall designate group bits by BG and channel bits by BE.

- for outputs, a single circuit allows address assignment.

Exterior connections are at the end of the card, as shown on the board in diagram 4.3. An AC/ signal indicates that the input has been recognized or that the output has been memorized. Negative logic is used in both inputs and outputs.

IV.2.2. Exchange of data :

All the data goes through the accumulator.
At input, two instructions are to be used in order to take full advantage of the system's capacity. The first instruction precharges the accumulator with the channel address, the second is the input instruction itself (including the group address).

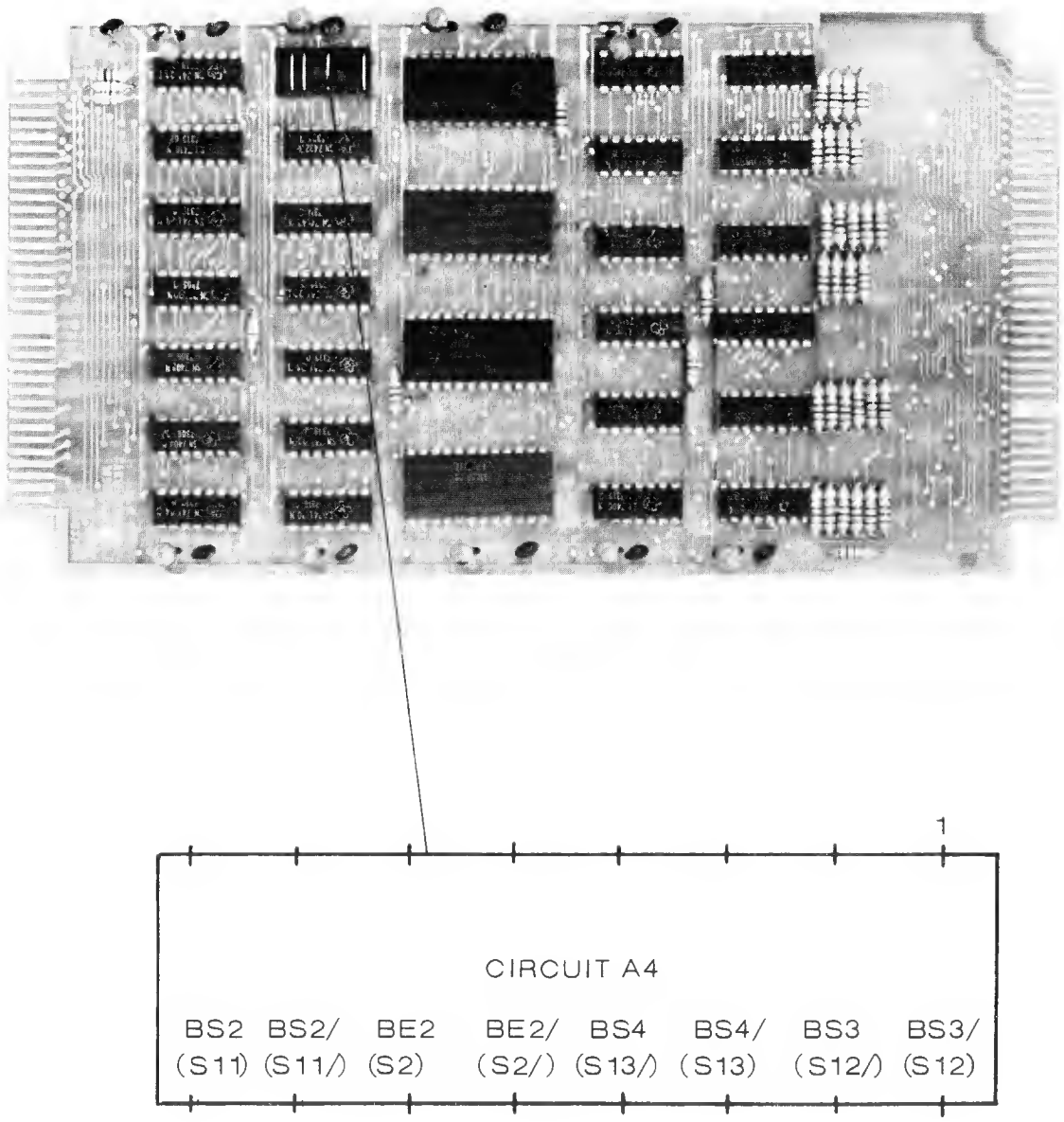
```
LAI  215 Precharge accumulator

INP  2   Instruct input.
```

Diagram 4.1.

The two instructions on diagram 4.1 address channel 215 of Group 2. The eight bits of this input are transferred at T'3 in the accumulator, through bits MD_0 to MD_7 of the pluribus.

Output is different. One instruction executes the transfer of the accumulator's content to the eight bits of the channel, through bits S_0/ to S_7/.



Assigned devices	BE	Input couplers
	BS	Output couplers
Number of devices assigned	BE	4 bytes or 32 inputs
	BS	4 bytes or 32 outputs

Figure 4.2 - 32 INPUTS 32 OUTPUTS ADDRESS ASSIGNED

IV.3. M1400 32 inputs 32 outputs interface

This card connects 4 input bytes (32 inputs) and 4 output bytes (32 outputs). MICRAL system cannot receive more than 2 cards of this type. On input, they can be used only on group 0's address. Therefore, it is not advisable to use this address as a stack channel. For input addresses, two configurations are possible. This is done by means of one single jumper which assigns 2 bits or the address register. Those two configurations are, respectively, channels 0, 1, 2, 3 and channels 4, 5, 6, 7.

For outputs, all configurations are acceptable except configuration 0-0 on the two highest bits, since this configuration is reserved to inputs. Diagram 4.2 shows the address assignments scheme on this card. BE for input bits and BS for output bits, have been used.

Exterior connections are performed at the end of the card, as shown on board in diagram 4.3.

An AC/Receipt signal indicates that the input has been recognized or that the input has been memorized. Negative logic is used for both inputs and outputs.

	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	AC/
E0	A28	A29	A30	A31	A32	A33	A34	A35	A36
E1	A19	A20	A21	A22	A23	A24	A25	A26	A27
E2	A10	A11	A12	A13	A14	A15	A16	A17	A18
E3	A1	A2	A3	A4	A5	A6	A7	A8	A9
S0	B1	B2	B3	B4	B5	B6	B7	B8	B9
S1	B10	B11	B12	B13	B14	B15	B16	B17	B18
S2	B19	B20	B21	B22	B23	B24	B25	B26	B27
S3	B28	B29	B30	B31	B32	B33	B34	B35	B36

Interrupt input : B37 Ground A37

Example = Bit 4 of input 3 must be connected at A4

INTERRUPT CONNECTIONS

BI/ = Bit d'interruption AINT/ = Annulation d'interruption.

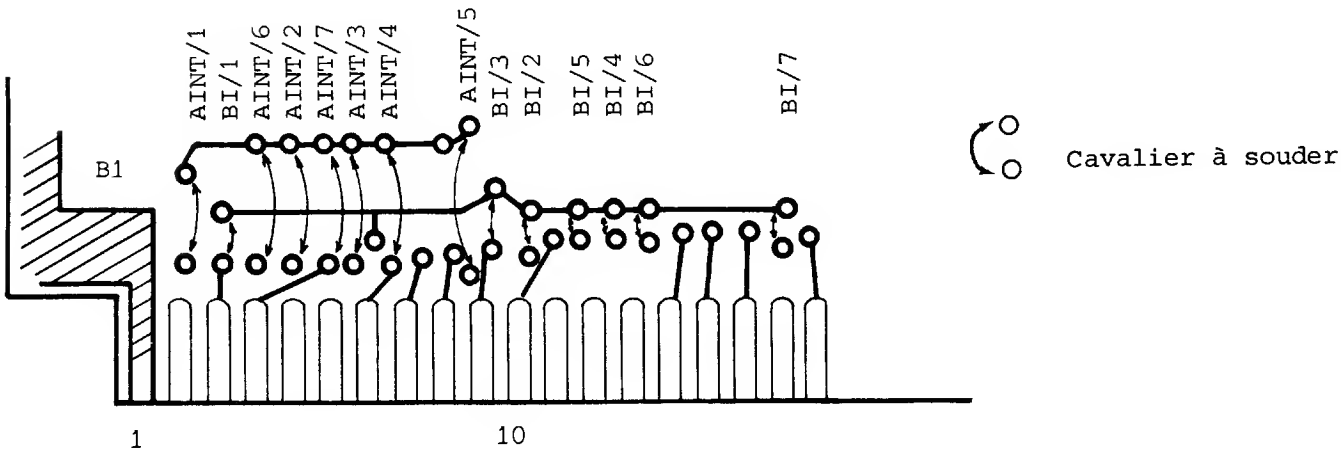
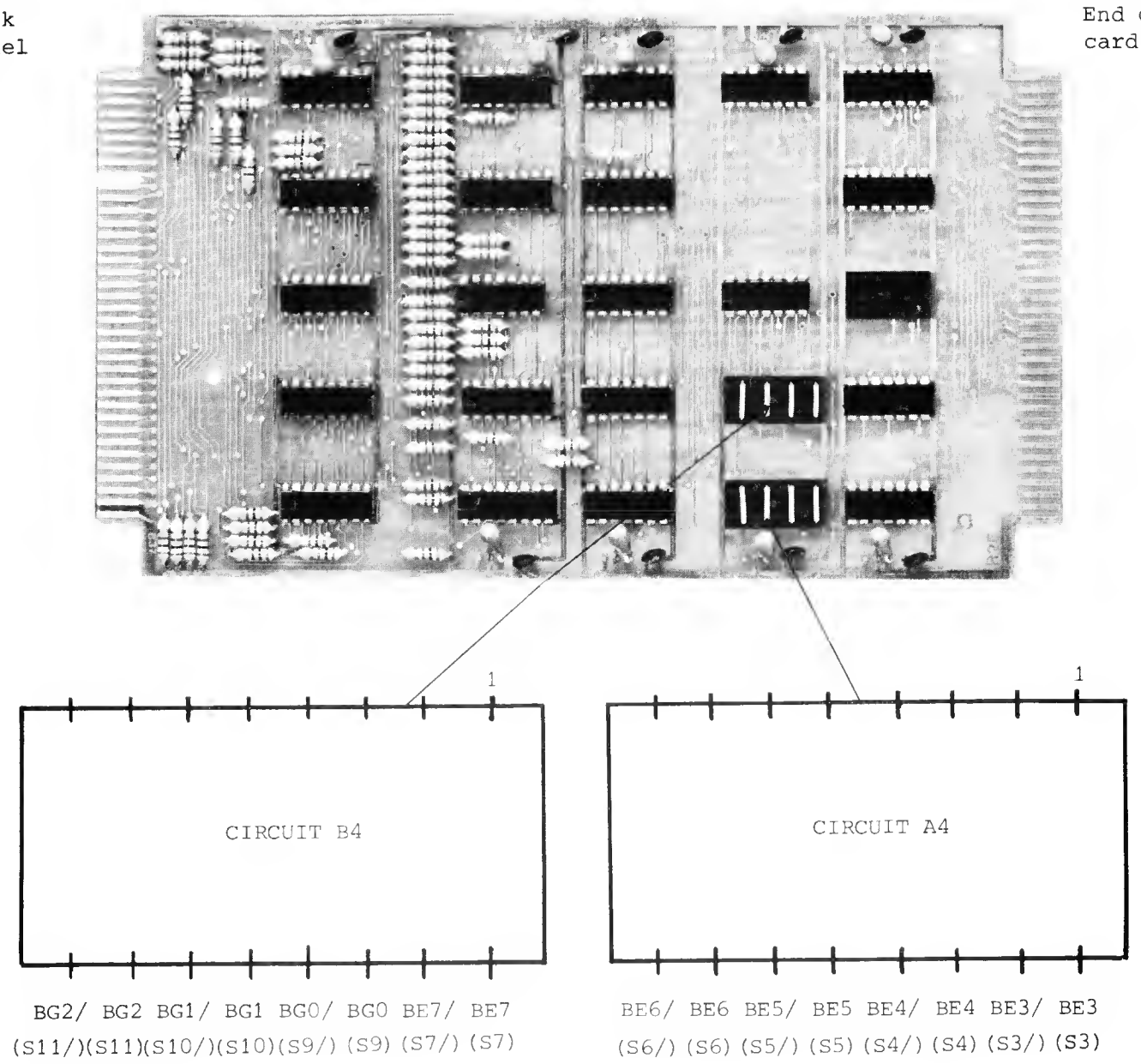


Diagram 4.3. Interconnection sketch of the card 32 inputs 32 outputs



Assigned devices : BG input group
BE inputs

Number of assigned devices : group : 1
inputs: 8

Figure 4.5 64 INPUTS CARD

IV.4. M1401 64 inputs interface

This interface allows a tie up with eight input bytes. Photograph 4.2. shows this card with its three address assignment circuits.

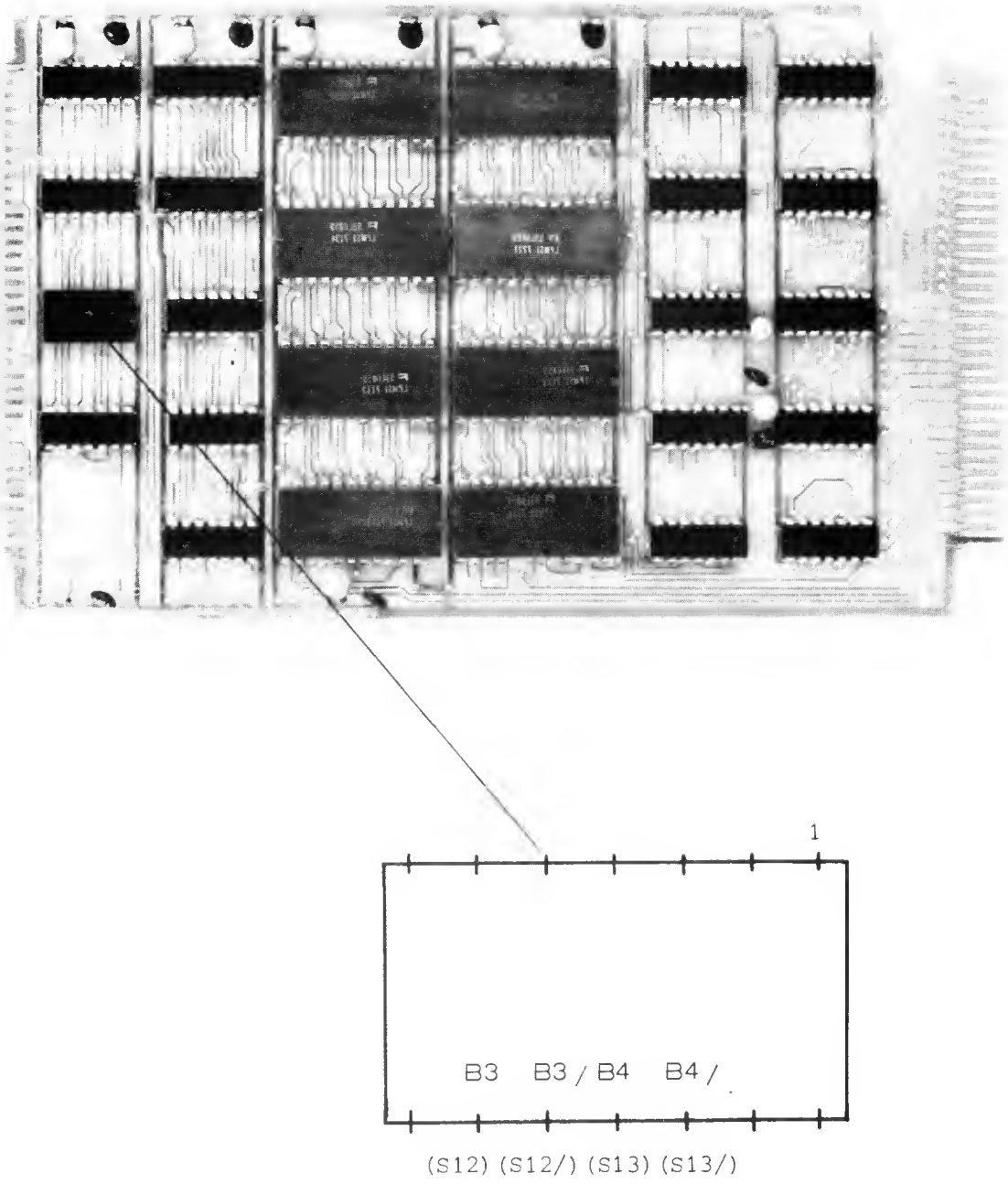
Circuit 1 assigns the group addressed by the INP instruction.

Circuits 2 and 3 assign the address of the channel designated by the accumulator content. Circuit 3 defines the two highest order bits, circuit 2 the three lower one (Diagram 4.4.).

Input of interfaces to the peripherals is done at end of the card, as per interconnection scheme shown on diagram 4.5.

	AC/	B7	B6	B5	B4	B3	B2	B1	B0
E_0	A9	A8	A7	A6	A5	A4	A3	A2	A1
E_1	B9	B8	B7	B6	B5	B4	B3	B2	B1
E_2	A18	A17	A16	A15	A14	A13	A12	A11	A10
E_3	B18	B17	B16	B15	B14	B13	B12	B11	B10
E_4	A27	A26	A25	A24	A23	A22	A21	A20	A19
E_5	B27	B27	B25	B24	B23	B22	B21	B20	B19
E_6	A36	A35	A34	A33	A32	A31	A30	A29	A28
E_7	B36	B35	B34	B33	B32	B31	B30	B29	B28

Diagram 4.5 - 64 inputs INTERCONNEXION DIAGRAM



Assigned devices — Outputs

Number of assigned devices — 8 bytes

B4/ B3/ is not allowed

Figure 4.6 - 64 OUTPUTS ASSIGNMENT

IV.5. M1402 64 outputs interface

It allows a tie up with 8"byte" outputs.

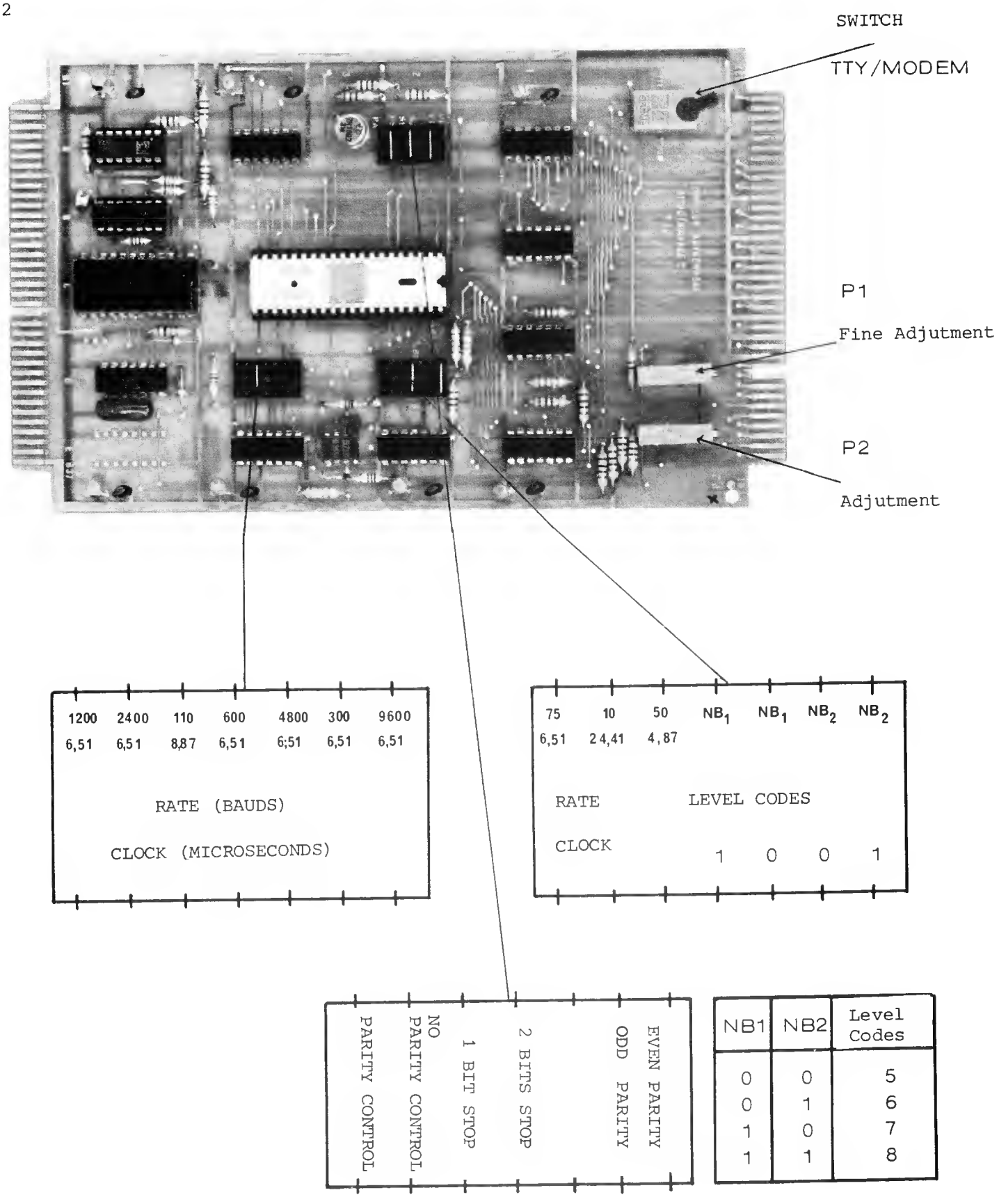
One single circuit allows address assignment since the maximal output capacity is 24 (Diagram 4.6.).

Care must be taken in output card address assignement to avoid 00 configuration on bits 4 and 5 (reserved for inputs).

As a matter of fact, any output card wired on basis of this configuration could not be recognized by the processor. Output of the interfaces on the peripheral side is performed at the end of the card. The interconnection layout is defined on diagram 4.7.

	AC/	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
S_0	A9	A8	A7	A6	A5	A4	A3	A2	A1
S_1	B9	B8	B7	B6	B5	B4	B3	B2	B1
S_2	A18	A17	A16	A15	A14	A13	A12	A11	A10
S_3	B18	B17	B16	B15	B14	B13	B12	B11	B10
S_4	A27	A26	A25	A24	A23	A22	A21	A20	A19
S_5	B27	B26	B25	B24	B23	B22	B21	B20	B19
S_6	A36	A35	A34	A33	A32	A31	A30	A29	A28
S_7	B36	B35	B34	B33	B32	B31	B30	B29	B28

Figure 4.7 - 64 OUTPUTS CARD INTERCONNEXIONS



NB1	NB2	Level Codes
0	0	5
0	1	6
1	0	7
1	1	8

Figure 4.8 JUMPER POSITION TO OBTAIN A DESIRED :

a) RATE
b) NUMBER OF CARACTER BITS
c) CONTROL MODE

(COMMON TO TTY AND MODEM)

IV.6. Universal asynchronous interface TTY

IV.6.1. Description :

This is a specialized interface. It only works from the pluribus power supply voltages. Several input/output interfaces are required to control and exchange data. The card controls either a teletype, or a MODEM.

The cards include the circuits enabling adjustment to the rate of transmission. This is obtained through a clock placed inside the card. Its rate is adjustable by means of two potentiometers P2 and P1 and suitable counters. P2 controls rough adjustment and P1 fine adjustment.

Circuits similar to those used for address assignment allow the selection pre-adjusted rates and to operate using different types.

IV.6.2. Standard output rates :

To adapt the circuit to a desired rate, two adjustments are necessary.

1°) Adjust the clock's period by means of P1 and P2 potentiometers and regulate the frequencies.

2°) Select the counters by a jumper.

Diagram 4.8 clearly indicates the jumper's position and the clock period for each of the standard values : 0 - 50 - 75 - 110 - 300 - 600 - 1200 - 2400 - 4800 - 9600 Bauds.

IV.6.3. Types :

IV.6.3.1.Number of Movements :

These can be chosen between 5, 6, 7 and 8 by using two jumpers as shown on the chart.

IV.6.3.2. Parity control :

Two jumpers allow, one to carry out parity control, the other to perform it on even or odd parity.

IV.6.3.3. Stop bits :

Finally, one last jumper allows to accept one or two STOP bits.

Diagram 4.9. gives the interconnection of the functions.

CONTROLS

VE/	INPUT STROBE	B9
AS/	OUTPUT STROBE	B37
PE/	INPUT READY	A29
PS/	OUTPUT READY	A6
SP/	FRAMING ERROR	A2
ES/	STOP BIT ERROR	A16
EP/	PARITY ERROR	A12
GROUND		B11-13-15-17-19- B21-23 A36-37

DATA

	B7/	B6/	B5/	B4/	B3/	B2/	B1/	B0/
INPUT	B1	B2	B3	B4	B5	B6	B7	B8
OUTPUT	A5	A1	A13	A17	A23	A19	A32	A28

Figure 4.10 Interconnexions common to TTY and MODEM

CONNEXIONS TTY

MOLEX CONNECTOR	CARD OUTPUT
5	B30
6	B27
7	B28
8	B29

CLOCK AJUSTMENT	: 8,87 MICROSECOND
RATE JUMPER	: 110 BAUDS - 5-10 D4
LEVEL	: 8 1-14 and 4-11 D2
STOP BITS	: Depends on options
PARITY	

Figure 4.11 ASR33 Interconnexion

RQS/	EMISSION REQUEST	B22
RQS	TO MODEM	B24
DTR/	TERMINAL READY (From Output)	B25
DTR	TO MODEM	B20
DSR/	MODEM READY FOR RECEPTION (To input)	A34
DSR	FROM MODEM	B12
CTS/	MODEM NOT READY FOR TRANSMISSION (To input)	A21
CTS	FROM MODEM	B14
CAR/	CARIER NOT DETECTED (To input)	A25
CAR	FROM MODEM	B16
MODEM DATA		
TDAT	TRANSMISSION TO MODEM	B18
RDAT	RECEPTION FROM MODEM	B10

JUMPERS POSITION : AS DESIRED

Figure 4.12 Modem Interconnexion

IV.6.4. TTY Mode description :

After the switch located on the card is positioned on TTY mode, the interface can be connected to a teletype. With a ASR 33 teletype, reader and punch control option must be used (the source current must 20 mA).

The connection is carried out on the connector board, which is on the back of the teletype, as per board in diagram 4.10.
To accept the information outputed in parallel, a pulse must be sent on to the received data enable. This can be obtained simply by connecting it to the recognition signal of the output interface which has been used.
When the information is accepted on an input, the input request must be acknowledged on the output recognition terminal.

The TTY, and MODEM, can be tested on the INPUT READY or OUTPUT READY terminal, in negative logic.
Those signals can be used also for interrupt.

IV.6.5. MODEM Functioning :

Diagram 4.11 indicates input and output functions of the MODEM.

TDAT transmitted data to MODEM

RQS emission request

DTR data terminal ready

RDAT data received from MODEM

DSR MODEM ready to receive

CTS MODEM not ready for transmission

CAR Carrier not detected.

Input strobe, output strobe, input ready, output ready signals, as well as the three Error signals, must be used as with type TTY.

IV.6.6. Error signals :

A framing error signal indicates either an emission or a reception request, while the previous INPUT data has not yet released to the peripherical, and the input or output are not ready.

A stop bit Error signal indicates that the number of STOP bits is not in accordance with the one that has been specified by using a specific jumper.

A parity ERROR signal indicates that parity is not the one specified by the corresponding jumper.

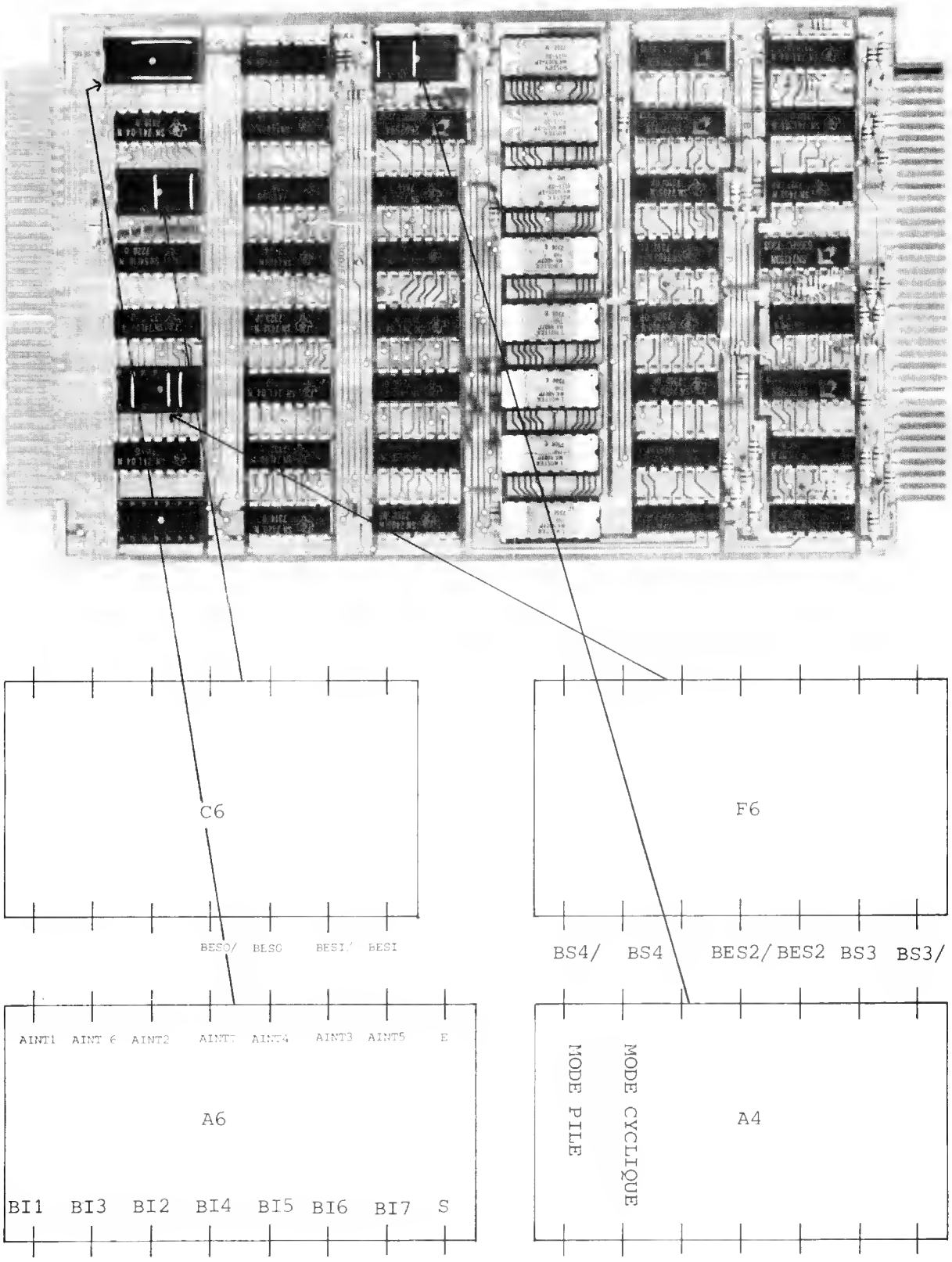


FIGURE 5.0 - CG CHANNEL STACK

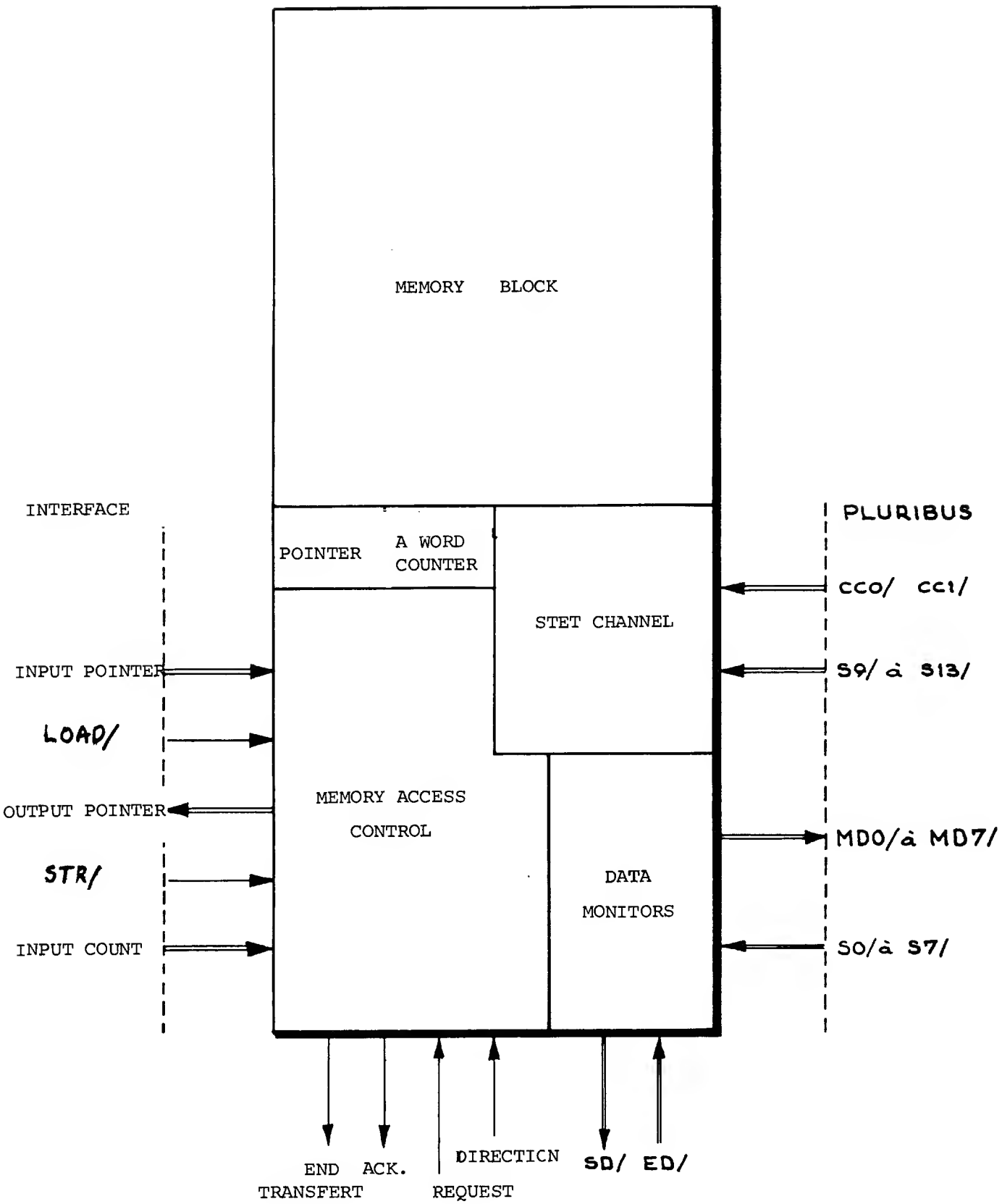


Fig. 5.1. STACK CHANNEL

V - Stack Channel Cards

V.1. General description of Stack Channel Cards (Diagram 5.1).

A stack channel card is made of a scratch pad memory with two multiplexed access :

- one to the processor is connected by terminals $S_0/$ to $S_7/$ in output and $MD_0/$ to $MD_7/$ in input.
- the other, to the exterior, is intended to connect a fast peripheral.

The latter is also composed of 8 bits in input and 8 bits in output.

The stack is composed of a random access memory addressed by a pointer updated automatically at each transfer of information. An input/output enables one, if desired, to read or write into the pointer register. They can be accessed through input and output interfaces. In the channel transfer mode, sequencing of the operations is performed by transfer request and receipt. An input controls the direction of the transfer. A word counter controls the performance by decreasing its content at each transfer and by stopping it automatically when the content is zero.

An input can precharge its content with a desired value. The channel transfer is initiated by loading the counter. The direction of the transfer can be controlled by the corresponding terminal. An OV signal on this output corresponds to"write"in the stack. A 5 V (or disconnected terminal) signal corresponds to "read" in the stack.

The end of the transfer, corresponding to zero on the word counter, generates an OV signal on the end of transfer output. This transfer can be utilized either through interrupt, or by test through an input interface. Upon input, only group addressing is used. It is therefore not necessary to precharge the accumulator to address the stack. This is done to increase access speed, but neutralizes the 256 inputs corresponding to the group address used.

In order to leave the possibility to utilize 32 input 32 output interfaces, which can only occupy group address 0, it is recommended not to use this address as a stack address.

```
LAI 25          PRECHARGE WITH DESIRED ADDRESS
OUT 15          WRITE INTO POINTER
LAI 47          PRECHARGE DATA
OUT 6           OUT PUT STACK
```

Figure 5.2

```
LAI 26          PRECHARGE HIGHEST BIT ADDRESS
OUT 15          WRITE INTO POINTER
INP 6           INPUT FROM STACK
```

Figure 5.3

```
LAI 127         LOWEST BIT ADDRESS PRECHARGE
OUT 15
LAI 3           HIGHEST BIT ADDRESS PRECHARGE
OUT 16
LAI 47          DATA
OUT 6           OUTPUT TO STACK
```

Figure 5.4

```
ORG /10
MAS             INTERRUPTS
OUT STAK        A  IN STACK
LAB
OUT STAK        B  IN STACK
LAC
OUT STAK        C  IN STACK
LAD
OUT STAK        D  IN STACK
LAE
OUT STAK        E  IN STACK
LAH
OUT STAK        H  IN STACK
LAL
OUT STAK        L  IN STACK
DMS             UNMASK INTERRUPTS
```

Figure 5.5

V.2. Functioning of the stack

V.2.1. Stack utilization, through processor control :

Two instructions can progress into the stack by input or output addressed to the stack. INP STCK decrements the checker and then read the stack at the pointer's address. One remark here : when one reads the stack after the pointer has been located, it is not read at the loaded address, but one unit below.

OUT STCK writes into the stack and then increments the pointer. An OUT STCK will write in the same memory location as the read by a previous INP instruction, provided no modification of the pointer occurred between the two instructions.

V.2.2. Access to the pointer and to the counter :

If access to the pointer is desired, it must be coupled through an input and an output interfaces. The accumulator must be precharged by the appropriate address : the desired address when a write instruction is to be done, the upper address when a read instruction is to be done.

An output instruction towards the corresponding interface's address then enables the transfer of the content of the accumulator into the pointer. One input or output instruction towards the stack then gives the desired reading or writing into the stack.
Diagram 5.2. shows a set of instructions allowing to write 47 (taken as a data) at address 25 of the stack (assigned at address 6), the pointer being connected to interface 15.
To read at the same address, the diagram's instructions should have been utilized.

Access to the counter is simpler : just transfer the precharged account in the accumulator towards the interface connected at the counter's input through an output instruction. This action activates the channel transfer process in the direction indicated by the corresponding order.
In the case of a 1024 byte stack, the pointer and the counter must be coupled throughout two byte inputs and two byte outputs.
For instance, the counter's loading instructions could look like diagram 5.3 (the counter's address for the high order bits being 15 and the counter's address for the low order bits being 16).

V.2.3. Utilization of the stack for saving register on interrupt :

Utilization of the stack is particularly interesting when an interrupt takes place. The registers and control indicators must be protected. The program counter in the processor's stack is protected by output instructions given to the stack address after a transfer in the accumulator by input instructions to the stack address.
To avoid being interrupted while protecting the registers, MAS must be used as first instruction of an interrupt program. Since the first instruction after an interrupt cannot be interrupted, this safeguards the registers without the risk of being interrupted.
Diagram 5.4. shows an example of a program safeguarding the registers corresponding to interrupt level 3. In that case, the control indicators are lost. If they have to be saved, they must be successively tested by conditional instructions without executing operations, increments and decrements to avoid modifying them.

V.2.4. Connection of a peripheral through the stack channel :

To connect a peripheral through a stack channel, a certainnumber of input/output interfaces must be provided for :

1°) one output to load the pointer;
2°) one input to read the pointer;
3°) one output to load the pointer;

The connections to the peripheral itself are the following :

1°) eight input wires of EDO/ to ED7/ data to be connected to the peripheral outputs;
2°) eight output wires of SDO/ to SD7/ data to be connected to the peripheral inputs;
3°) one input wire connected to the exchange request of the peripheral (DE/);
4°) one exchange receipt wire;
5°) one wire reporting the block transfer end FTB, which can be connected to an interrupt input as indicated in paragraph II.3.2.1.

Actually, it is possible to connect up to 10 peripherals to one single stack channel provided one is disposing of output interfaces designed to switch on the desired peripheral. In that case, only one peripheral will be active. The maximum of 10 is due to the FAN out of the circuits.

Control of the stack through the processor has priority on the automatic transfer mode. That means that each input or output directed to the stack address will set the counter to zero.
To resume the transfer, the counter must be restarted.

V.2.5. Shape of signals :

Note that the connections to the pointer are optional and are necessary only if random access to the stack is desired.

The exchange request and receipt can be performed in two different ways :

a) if the channel has been activated prior to any exchange request, the request may be a pulse less than 0,500 µ.sec. wide. Receipt then appears as a pulse of 0,6 microsecond coming at the latest 0,5 microsecond after exchange is requested;

b) if the channel has not been activated, the request appears in the form of a level which is reset when the first discharge signal shows up. Negative logic is used. To load properly the pointer and the counter, they must be verified by means of the receipt output of the corresponding interface on LOAD/inputs for the pointer and STR/ for the counter.

CHANNEL STACK CONNEXIONS

1°) With other channels

LINK IN	B11
LINK OUT	B2

2°) With peripherals

DT/	Transfer signal	B17
BT/	Transfer Flip Flop	B4
STDO/	Data out strobe	B3
IN/OUT/	Transfer direction	A1
CD/ (0 à 7)	Data output	voir tableau
CS/ (0 à 7)	Data input	voir tableau

3°) Through input/output ports

TC/	Cyclic transfer	A35
PA/ (0 à 7)	Address precharge	voir tableau
LOAD/	Address strobe	B29
A/ (0 à 7)	Pointer address	voir tableau
PR/ (0 à 7)	Programmation long block	voir tableau
STPC/	Counter strobe	A12
BTB/	End of Block signal	A10 B10

4°) Ground A36 et A37

	B7	B6	B5	B4	B3	B2	B1	B0	Strobe
CD0/ à CD7/	A30	A31	B27	B22	B12	A11	A9	A8	B3
CS0/ à CS7/	B6	A5	B8	A6	B7	A7	A3	B5	B17
PA0/ à PA7/	B30	B31	B32	B33	B34	B35	B36	B37	B29
A0 à A7/	A21	A22	A23	A24	A25	A26	A27	A28	
PR0/ à PR7/	A13	A14	A15	A16	A17	A18	A19	A20	A12

FIGURE 5.7.

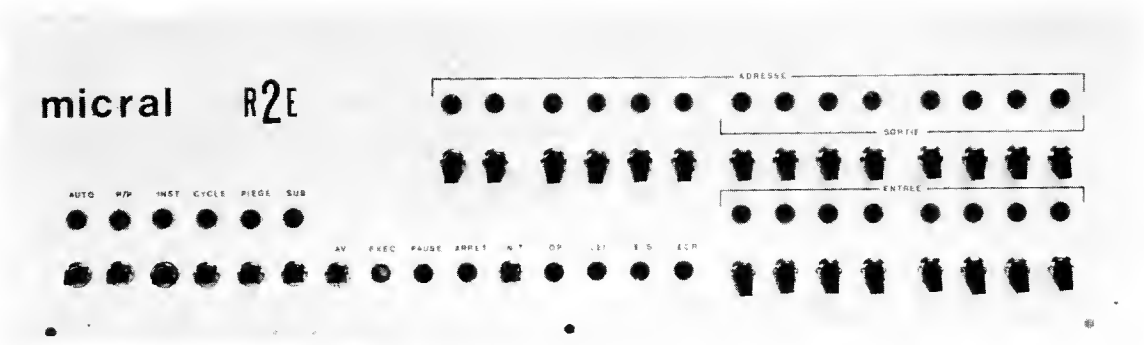


PHOTO 6.1 FRONT PANNEL CONSOLE

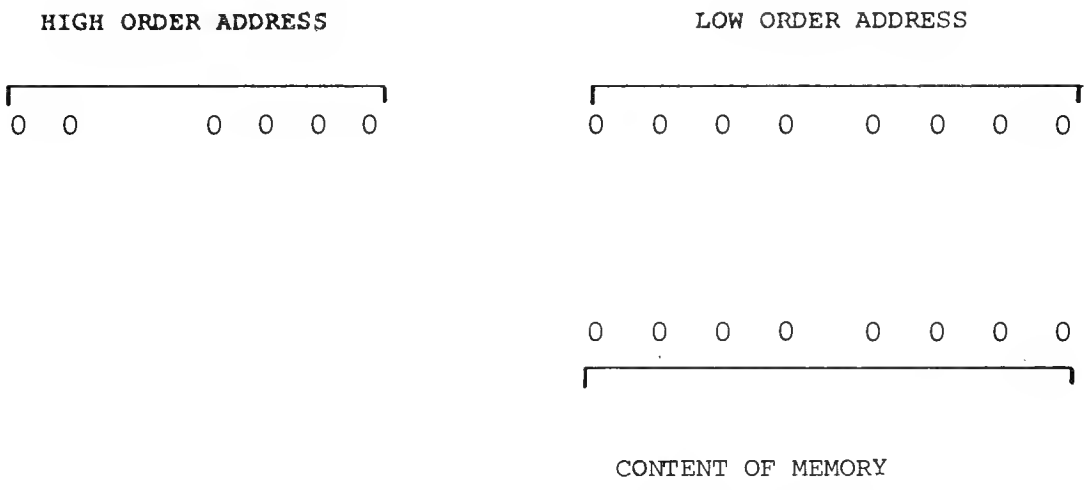


Figure 6.1 - EXPLANATION OF OP CYCLE LIGHTS

(FOR READ CYCLE SAME CONFIGURATION)

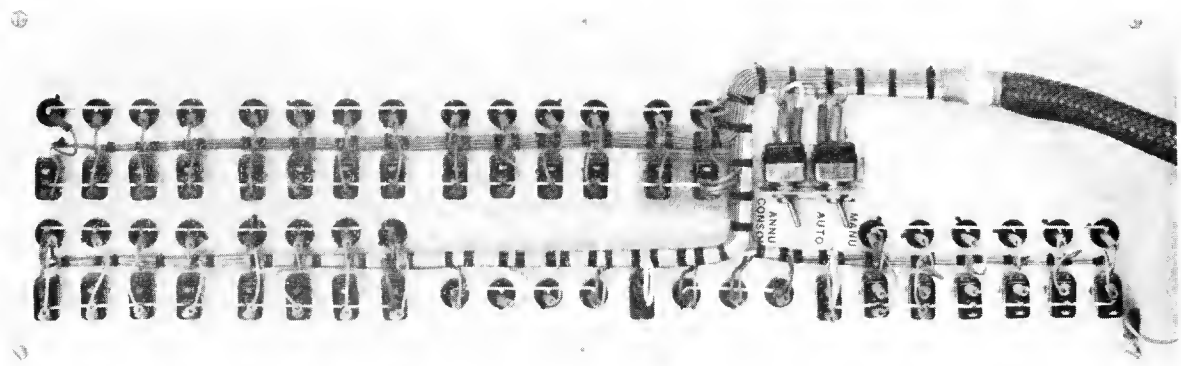


PHOTO 6.2 FRONT PANNEL REAR VIEW

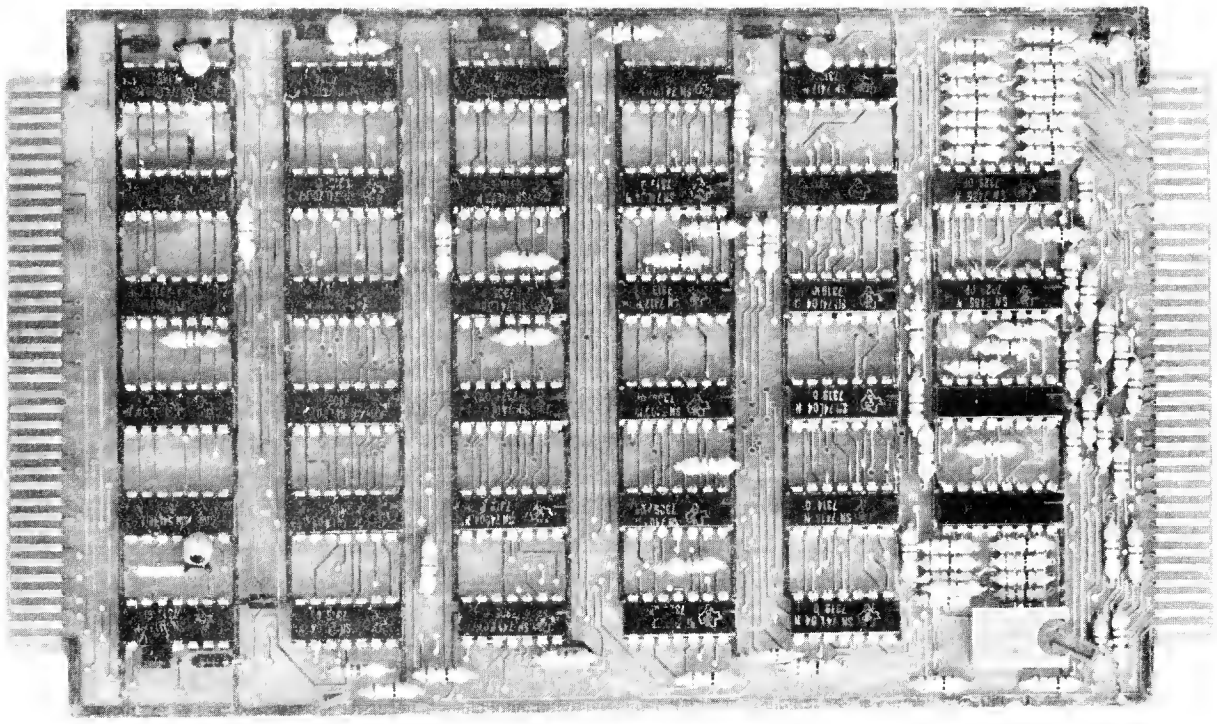


PHOTO 6.3 CONSOL CARD

VI - Visualization console

VI.1. General principles concerning visualization :

MICRAL's principal use is in process control. It does not aim to be an universal mini-computer. For this reason, the console is not supplied in the standard version.

Generally speaking,when MICRAL is utilized within a system, a console matching the particular application will have to be designed. The simpler it is, the easier it can be used by people who are not familiar with data processing. Just a signal, indicating a normal RUN, will be often sufficient. The optional console of MICRAL (photo 6.1) is actually meant for studies, adjustment, servicing and for systems handled by specialized personnel. The console enables them to command the mode with which MICRAL will work, as well as to visualize certain information needed for during use.

VI.2. Modes :

MICRAL, equipped with the console, can run under five modes :

- automatic;
- stepping;
- traps;
- Substitution;
- servicing.

VI.2.1. Automatic mode :

It is the normal mode of operation. In this mode, programing instructions proceed normally and can be stopped only by lack of low voltage on READY/terminal of the pluribus and by a HLT instruction. To get the automatic mode, depress fugitive key AUTO and at the same time make sure that signal TRAP is out. If it is lit, press on key TRAP to put it out.

RUN lightdisplay must be on WAIT and the stop light off, indicating that the program proceeds normally.
If signal RUN goes off, one of the signals WAIT or STOP must keep alight indicating to the operator in what state the system is. In the WAIT state, it indicates that the program has presented a HLT instruction.

By examining the lights, whose sequence is determined by the processor, one can interpret those memory areas which are the most requested. In the STOP state, the lights marked address indicate which address the program has stopped at. The eight right hand lights display the lower order address, the six left hand display the higher address, which is the page address.

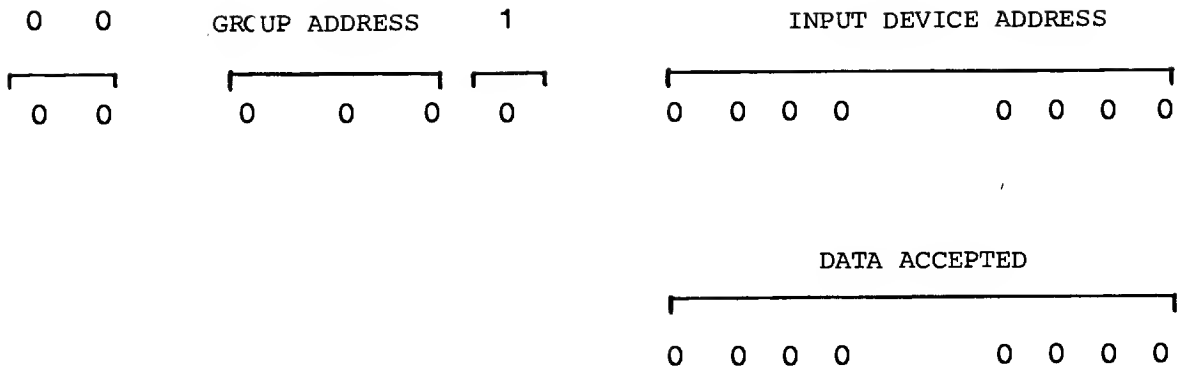


Figure 6.2 EXPLANATION OF i/o INPUT CYCLE LIGHTS

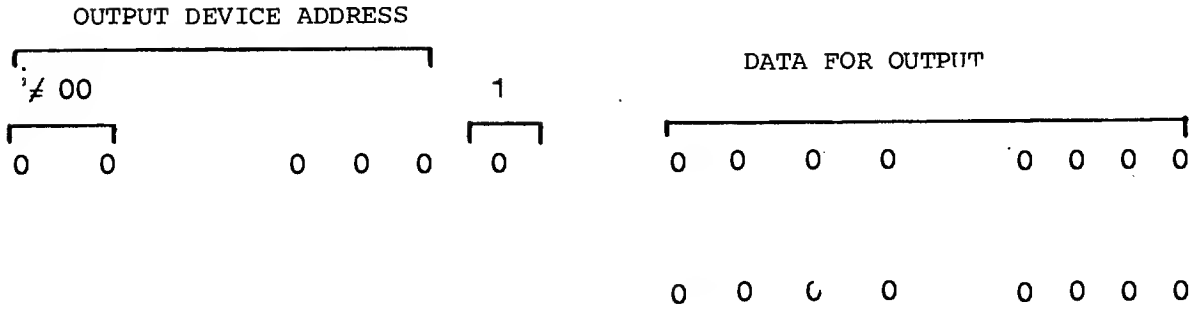


Figure 6.3 EXPLANATION OF i/o OUTPUT CYCLE LIGHTS

The eight lights marked INPUTS display the content of the addressed memory or the data supplied by an input interface.
In the WAIT state, the cycle lights (FETCH READ I/O WRT) indicates the cause for the WAIT state : the memory is at fault when one of the lights is lit. A peripheral is at fault when I/O light is on. It is particularly the case when a wrong address is concerned. The meaning of the signal will be analyzed in the paragraph describing the stepping mode.

VI.2.2. Stepping mode :

To switch to stepping mode, depress fugitive "step" key. The RUN light goes out. The step light comes on. The stepping mode provides two options :

- instruction;
- cycle.

In the first one, the processor is set in WAIT state at each instruction on FETCH cycle. To get this option, depress fugitive key INS (the corresponding light comes on).

In the cycle option, the processor is set in WAIT state at each memory cycle.

In both cases, stepping is obtained by pushing STEP key.
Clarification of the signals in WAIT state is given in following paragraphs and diagrams 6.1 - 6.2 - 6.3.

In the INS position, when at least one step is made, the ADDRESS lights display the instruction address to be executed at the next step and the INPUT lights display instructions to be executed, thus allowing the control execution of the program.

In the cycle position, interpretation of the lights depends upon the cycle lights :

- in OP, the meaning is the same as in previous paragraph;
- in READ or WRITE, the memory address can be read;
- in READ, the input signals display the data delivered to the processor by the memory;
- in I/O, the left hand bits of the address display the six lowest order bits of the instruction to be executed. The two highest order bits report that the instruction is an input when they are in OO pattern and that the instruction is an output when the pattern differs from OO.

On input instructions, the three following bits display the group address. The eignt bits of the lower order address display the input address. The data to be noted is displayed on the input lights.

On output instructions, lowest order bits of the higher address indicate the peripheral's address. The eight bits of the lower address, marked OUTPUT, indicate the data to be exchanged.

To get out of the stepping mode, depress the desired mode key.

VI.2.3. Trap mode :

In the TRAP mode and in RUN, the program will proceed normally until the ADDRESS lights and the keys below these lights present the same pattern. When coincidence occurs, the processor goes automatically in stepping mode and a synchronization pulse is generated. Coincidence can occur only in OP cycle, thus enabling the program to be stopped at a desired FETCH address.

To switch to TRAP mode, depress the fugitive key (the corresponding light comes on).
After coincidence has occured, stepping can be performed by using step key. As soon as AUTO key is depressed, the processor will carry on normally until a new coincidence occurs.
To get out of TRAP mode, depress again TRAP Key. TRAP mode can be used in a different manner : by disconnecting the two points marked A and B on photograph 6.1 representing the console, the program does not stop any longer at the coincidence, but generates a pulse on the TEST terminal allowing the synchronization of an oscilloscope.

VI.2.4. Substitution mode :

In this mode, the configuration displayed by the INPUT marked keys is substituted to the memory output, enabling it to enter a program from the console, in stepping mode.

VI.2.5. Repair mode :

In this mode, there are several possibilities :

1°) by means of a switch located on the console, an automatic substitution mode can be obtained in which the READY/terminal is automatically fed with a low voltage current to prevent WAIT state.

The instruction displayed by the keys is processed indefinitively so observation can be made easily;

2°) a switch placed in the back of the console (diagram 6.2.) prevents the execution of a zero instruction when the power supply starts up. In that case, the processor sets itself immediately in stepping mode, enabling, through substitution mode, to enter at any address of the program without executing the instruction of the address.

VI.3. Putting the console out of service :

This is effected by depressing the second switch at the back of the console (see photograph 6.2.).

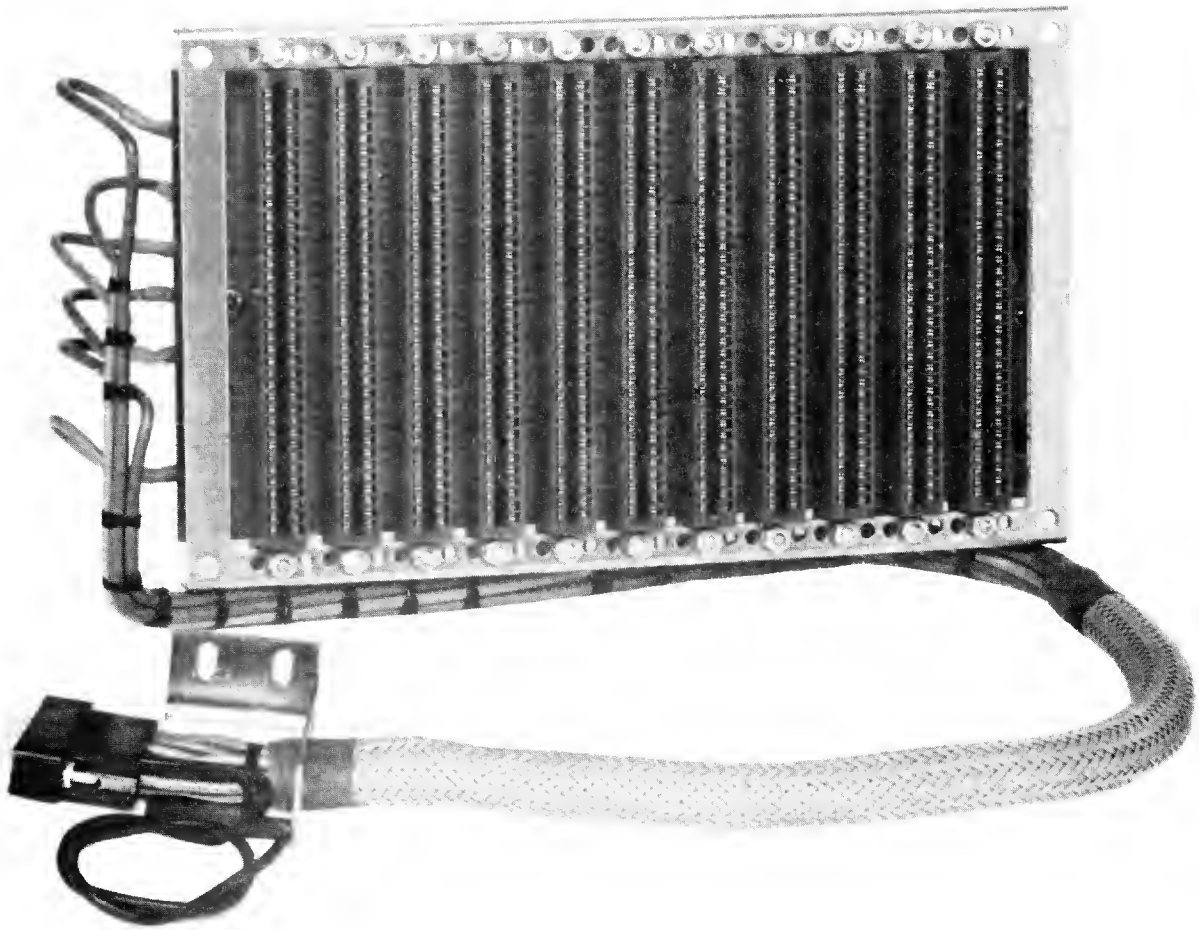


PHOTO VII.1 PLURIBUS

VII - The pluribus

VII.1. General description :

The pluribus performs two functions :

1°) materially, supports the cards that perform the various functions;

2°) electrically, connects the cards one to another.

Electrically, the pluribus includes 74 terminals of which 9 are unused.

Annex 1 lists terminals with their functions.

Fuctions are broken down as follows :

- Power supplies
- Control signals
- Addressing signals
- Data

Clarification for each function is given in the description of the cards where they are generated and used.

A general descriptive paragraph is given in the technical appendix.

VII.2. Functional description of the terminal :

VII.2.1. Power supplies :

Ground	A/	A37	Max. Current
+ 5 V	B/	B37	1 A
- 9 V	B2		1 A

Memory protect voltage.

The emergency terminal enables the MICRAL to maintain a voltage on the RAMS so as to avoid losing information upon mains supply failure. It is not supplied in the standard version.

VII.2.2. Synchronization :

All these signals are generated by the processor and utilized on all the other cards.

Φ_2 A8

It is the time base phase used to determine precisely all MICRAL's functions. From the processor a viewpoint, two Φ_2 can be used : the first in phase with SYNC designated by Φ21, the other in phase with SYNC designated by Φ22.

SYNC/ B23

Defines the basic period of the processor.

T_2/ - A27

The signal relates to the addressing period of the highest order of the memory address, of the group address in the input instruction and of the address of the output peripheral.

T3/ - A35

Gives the state of the processor which immediately follows T2 state or WAIT state.

T'3 A28

Comes from the processor and is used in all the cards. This timing is the most important under the user's viewpoint. The MICRAL depends upon this timing to monitor information correctly. It must come up no later than 500 nanoseconds from the end of T2, and have covered T3.

If the data appears too late on the bus, there is a chance that the processor will take wrong information into account. To avoid this simply, do not apply READY/ until the data is established. Then, no risk of taking wrong information into account.

CC0/ A30

CC1 A31

Those two bits appear during the T2 state and define the memory cycle of the current phase. The diagram in the chapter describing the processing card shows the equivalences.

WAIT / B35

Acknowledges that the processor is in the WAIT state due to the fact that READY/INSTRUCTION has not returned to zero fast enough.

STOP / A36

Indicates that the processor is in the STOP state following a HLT instruction. Reactivation of the program can be obtained only by a MRST signal or by an interrupt.

VII.2.3. Instructions :

READY / B9

When this terminal is at + 5 V, the processor switches to a WAIT state after T2 state. If the information requested by the processor is available early enough to avoid WAIT state, the addressed device must send back a low voltage current to this terminal at the latest 500 nanoseconds after T2 falling edge.

READY C B6

This function is created by the console to obtain stepping mode. When a low voltage charge is applied to this terminal, the processor switches to WAIT state even if READY/ comes back to OV. As soon as it reaches + 5 V, execution proceeds normally.

INIT / B28

When an OV is applied to this terminal, the processor is activated again. This voltage commands a RAZI control which itself creates a zero level interrupt. The duration of INIT must be above 3 nanoseconds.

RAZI / B8

This control switches to O on INIT command and comes back to + 5 V upon receipt of the zero level instruction.

SUB / BII

This voltage, created by a console key, inhibits all the cards' outputs towards the processor and enables the console's keys to force through an instruction, in stepping mode.

V..2.4. Address :

S_7/ A 20 S_6/ A 21 S_5/ A 22 S_4/ A 11 S_3/ B 15 S_2/ A 16

S_1/ B 17 S_0/ B 16

On these terminals is located the lower address of the memory addressed in the OP, READ and WR cycles. In the I/O cycle, one finds the sub-code of the input address, or the output data.

S_{13}/ B 31 S_{12}/ B 34 S_{11}/ A 33

S_{10}/ B 33 S_9 / A 26 S_8 / B 26

On these terminals are located both the address of the memory page addressed to in cycles OP, READ and WRITE, and the six low order bits of the instruction code in cycle I/O (the highest order bits being CC0 and CC1).

Where an output is concerned, bits S_{11}/ S_{10}/ S_9/ specify the group address.

Where an output is concerned, bits S_{13}/ to S_9/ specify the peripheral address.

Note that all codes appear in negative logic, a bit 0 will show itself as a high voltage.

VII.2.5. Data :

$D_{7/}$ B 30 D_6/ A 29 D_5/ A 24 D_4/ B 24

D_3/ A 18 D_2/ A 15 D_1/ A 17 D_0/ B 27

The data addressed to the memory are placed on these terminals. They appear during T'3 state. As the latter is actually evolded from T2 and not T3, this datum is available when the system is in the WAIT state, in stepping mode. It will therefore be possible to control what is written into memory during the end of the cycle.

MD_7/ B 25 MD_6/ A 25 MD_5/ A 23 MD_4/ B 19

MD_3/ A 32 MD_2/ B 29 MD_1/ A 6 MD_0/ A 34

The data destined to the processor are located on these terminals. They come either from the memory, or from the input interface, or from the input keys of the console. Same as per the previous paragraph regarding its presence during stepping mode.

S_7/ to S_0/ (see paragraph = addressing).

On these terminals the data regarding the output interface.

BI_7/ A 19 BI_6/ A 14 BI_5/ A 12 BI_4/ A 13

BI_3/ B 10 BI_2/ A 10 BI_1/ B 3

These terminals are meant to receive the interrupt level 7 to 1 respectively. Level 0 interruption must be applied to INIT/Terminal.

A INT_7/ B 4 A INT_6/ A 3 A INT_5/ A 9 A INT_4/ B 7

A INT_3/ A 7 A INT_2/ A 4 A INT_1/ A 2

Receipt of level 7 to 1 interruptions is on these terminals. Level 0 receipt is internal to the processor and is achieved by the rising edge of terminal RGZ1.

Very important note :

Concerning both inputs and outputs, all signals followed by the sign / are present or must be present in negative logic. In other words, a high voltage corresponds to "0" and a low voltage to "1".

S O F T W A R E

I - Introduction

MICRAL is the first of a new generation of mini-computer whose principal feature is its very low cost.

a) Low Hardware cost resulting from :

1°) use of integrated micro-processor (MOS-LSI circuits);

2°) modular structure allowing minimal HARDWARE configuration for a given system;

3°) no need of loading peripheral due to the use of Read Only Memories.

b) Low operating cost through use of non-specialized personnel :

The above automatically implies, in the system's basic design, complete integration between SOFTWARE and HARDWARE.

I.2. Stages of a system design :

The stages of a system study will in most cases be the following :

1°) Hardware configuration design :

- definition of the standard interfaces;
- examination of the specialized electronic (by that we mean electronics adapted to a particular system that cannot be built from of standard components).

2°) Software evaluation :

This step consists of estimating the size of the memories by category (ROM - RAM).

3°) Cost evaluation :

- Hardware
- Software
- Firmware — ROM Hardware cost
  ROM programing cost

4°) Design

4.1. Program coding
4.2. Program assembling
4.3. Program simulation and debugging
4.4. Hardware design
4.5. System debugging
4.6. PROM programing

It will also be possible to program the ROM, or certain parts of it, containing programs that one can be certain of earlier than indicated (4.6.), if one is disposing of study Hardware on exhausted configuration and providind the address assignment possibilities are flexible enough within the programing phase.

As a matter of fact, in the initial phase, it is recommended to simulate the RAM program, thus giving the possibility of loading the program pretty frequently. The MICRAL has, of course, an optional 5 V permitting the information to be returned in the event of mains failure.

When a sub-program is set-up and when one does not want to loading problems and risks of information losses, one can program it in RAM.

As it is impossible to mix RAM and ROM, one must modify the programs of the addresses which one wants to put into ROM.

I.3. Study and working configuration :

Therefore, it is important for the programmer not to mix up study configuration and working configuration.

To optimize study costs, we must dispose of an exhausted configuration, thus avoiding time wasting. In operation, configuration should be reduced as much as possible.

II - MIC 01 MICRAL MONITOR

II.1. Introduction :

MICRAL MONITOR is stored in ROM in the highest addresses of the memory. It is intended for program debugging.

It allows :

1°) to load programs stored on binary tapes RAM outputed by the assembler or the MONITOR;

2°) to execute programs stored in memory starting at a specified address with the possibility to hold program instructions at a desired address;

3°) to display and modify Registers and control flags;

4°) to dump memory between two specified addresses;

5°) to display and modify consecutive memories starting at a specified address;

6°) to create a binary tape from a program stored in memory between two specified addresses. This tape can be directly loaded by the correct monitor command;

7°) to create a Tape for PROM programming.

II.2. Hardware configuration :

The following cards are necessary to take full advantage of MIC 01 Monitor program.

11 2 1 Processor

Processor can be used with or without Real time clock.

11 2 2 Memory cards

MIC 01 MONITOR is stored in ROM between 3800 to 3FFF addresses. It uses also RAMS between 7E0 and 7FF.
Only one level of the internal stack is used.

11 2 3 Stack channel cards

One Stack channel cards is required. Its addresses must be 6 for input and 22 for output.

11 2 4 Universal Asynchronous Interface

This card must be connected as indicated in the HARDWARE chapter (paragraph IV 6.3, diagrams 4.9 and 4.10).
Data input must be connected through an output interface and output through an input interface, both being assigned to address zero.
Another input interface is used to control the status of the TTY as follows :

II.2.4. Universal asynchronous card:

The universal asynchronous card must be used as indicated in paragraph IV.6.3 of the Hardware chapter, diagrams 4.9 and 4.10. Data input is connected to an output interface and data output is connected to an input interface. The two interfaces must be assigned to address 0.

The second input interface must be connected to the card's command output as follows :

VE/ input strobe B9

AS/ output strobe B39

PE/ input ready A29

PS/ output ready A6

SP/ framing error A2

ES/ stop bit error A/6

EP/ parity error A/2

II.2.5. Interfaces +

The 32 input 32 output interface gives all the required connections. The first input address, as well as output address, must be zero. The connections as described in previous paragraph must be :

Input 0	TTY data outputs
Input 1	Teletype controls
Input 2	Stack pointer output
Output 0	TTY data inputs
Output 1	Teletype instructions
Output 2	Stack pointer input.

II.2.6. Teletype

The teletype used must be an ASR 33 teletype with either : punch control or tape reader control.

II.2.7. Visualization console :

This one is necessary for assembling the unit.

+ Using of the 32 input 32 output interface implies group 0 assignment.

INPUT READY ON BIT 0

STOP BIT ERROR on BIT 4

PARITY ERROR on BIT 5

FRAMING ERROR on BIT 6

OUTPUT READY on BIT 7

11 2 5 Interface

A 32 inputs 32 outputs interface is used. To summarize interface address assignments, the situation is as follows :

DATA FROM TTY : input 0
DATA TO TTY : output 0

TTY Status : input 1

11 2 6 TTY

ASR 33 must be used. The following options are required :

Punch control
Reader control
20 mA source current

11 2 7 Front Pannel Console

Front Pannel option is required to take advantage of the MONITOR as well as the ASSEMBLER programs.

11 3 MIC 01 Monitor Procedure

11 3 1 Introduction

MIC 01 is stored in the highest memory addresses, leaving the lowest memory addresses for the user's program storage. In order to enter the program, the processor should be placed in substitution mode and a jump to the monitor entry made with the console keys.

After mains supply voltage has been applied, address 3B30 should be used. This activates all memories used by the program. After this has been done once, the regular entry address is 3B49.

Check first that the TTY is on "LIGN" mode and switch the power on while depressing the step key to enter the wait mode. Depress then the SUB key to enter the substitution mode and use the pannel keys 44, 30, 3B successively.

The program will then be connected to address 3B30 by depressing the AUTO switch.

II.3.2. Instructions :

The following instructions can be used :

M : Memory display and modification
A : Register display and modification
H : H and L display and modification
I : Control and Registers display and modification
D : Memory Dump
G : Program execution
C : Binary tape leader generation
P : Binary tape punch
F : Binary tape trailer generation
N : ROM programming tape punch
R : Binary tape read.

Each instruction, typed on the Keyboard is followed by a colon. Except for R/ instruction, the user must then type control characters which specify and confirm the instruction.

When the Monitor does not identify a character, a question mark is printed and a new instruction must be typed. This can be used to cancel instruction.

II.3.3. M : DISPLAY AND MODIFICATION

The colon must be followed by an address and a carriage return, a space or a comma.
When the address includes less than four hexadecimal characters, zeros are filled in the higher order figures.
When the address includes more than four hexadecimal characters, a question mark is printed and a new instruction mus be typed.
A two hexadecimal character figure, followed by a dash (-), displays the value of the specified memory.

Generally speaking, a dash means that the value displayed can be modified by typing a new value. When the value typed includes more than two hexadecimal characters, the last two are accepted; a zero is shown if the value includes a single hexadecimal character.

Confirmation of the new value is obtained by typing a space or a comma; the following memory is then displayed and it is possible to continue from one address to next up to and unknown character or a carriage return which cancels the instruction.
Figure 11 I shows how to use M instruction.
For instance, M instruction can be used as soon as the program has been connected to the Monitor, to allow an easier entry.
Starting at address zero, the following values should be successively used : CO, 44, 49, 3B.

When this has been done, it is possible to enter the program by depressing the INIT key.

II.3.2.3. Registers Display and Modification :

Typing A allows the display and modification A, B, C, D, E, H and L successively. Typing H displays only H and L. Typing T displays controls C, Z, S and P followed by Registers.
Procedure is the same that with M. Figure II 1 shows how to use these instructions.

II.3.2.4. Memory Dump :

Typing D gives a Memory Dump. The instruction must be specified and confirmed by a comma and terminated by a carriage return. Memory values are displayed by sixteen two characters figure in a such way that each figure is in a column corresponding to the low order character. The first column is used to display the address of the memory displayed in the second column which includes the value of the memory whose lowest address character is zero, and so on.

In the first line, double dots fill the space in the memories not required by the instruction (for example, if the first memory to be displayed is 3024, space for memories 3020 to 3023 will be filled with double dots).
Besides, a space is provided between lines to allow annotation.

II.3.2.5. Execution of program stored in RAMS :

Typing G followed by an address connects the program to the specified address. The address can be followed by two other addresses providing two holds. The specified addresses must differ from, at least, 3.
Figure 11 4 shows how to use G instruction.

II.3.2.6. CREATION of a binary tape

This is done in three steps :

1) Leader generation

2) Tape generation

3) Trailer generation

The first one is obtained by typing C instruction followed by a carriage return.

The second one by specifing the first and the last addresses of the memory to be punched, followed by a carriage return (specified addresses must be separated by a comma or a blank).

The third one is obtained by an F instruction followed by a carriage return.

As many P instructions as desired can be typed between one leader and one trailer. Starting each bloc, bloc size and initial address are punched. At each bloc end, a check sum computed is punched giving a check when tape is read. Each byte is split in two and punched by two characters.

Figure 11 - 2 gives an example.

II.3.2.7. ROM programming tape :

Typing N followed by starting and ending address and confirmed by a carriage return generates a tape at the ROM programming format. (see Introduction, paragraph 1-4).

Figure 11-2 gives an example.

III - ASMIC 01 MICRAL ASSEMBLER

III.1. Introduction :

The ASMIC 01 assembler enables, the object program in machine code to be generated, starting from the source program. The assembling is obtained in two passes. The first pass generates the object program, the second generates a complete listing.
The source program is supplied in the form of a punched tape. The object program is delivered either by the TTY punch in the form of a binary tape, or in the form of a complete listing.
The assembler can be stored either in RAM or in ROM.

When used in RAM, it will have to be loaded as many times as necessary by using the monitor. When the assembler is stored in ROM, it will not be necessary to load it. It can be activated with the console in substitution mode by a jump instruction at the address of the entry.

III.2. Hardware configuration :

The Hardware configuration needed to use the assembler is the same as the one needed to use the monitor, except that the memory can be composed either of RAM or ROM, as required.
45 byte of ROM or RAM plus twice the space of the assembled program in RAM is required for the assembler.

III.3. Source program

III.3.1. Creation of the Source paper tape:

The source tape can be created from any Punch tape-keyboard set generating eight moment perforation in ASC II code. For instance, ASR 33 TTY in local mode. The user must punch his program by using the symbolic codes as described in paragraph II.2. of the Hardware chapter. Some pseudo-operations and special assembler symbols will be described below.

The formats required are described in the following paragraphs.

III.3.2. Program lines :

The assembler takes into account the source tape line by line. Two lines are separated by the following characters' group :

CR Carriage return

LF Line feed

RO Rub-out signal

RO Rub-out signal

These characters must be presented in the same order.

Two types of lines are accepted :

a) comment line,

b) instruction line.

When the first character of the line is an asterisk, this line will be considered by the assembler as a commentary.
In the opposite case, it will be considered as an instruction line.

The character @ that can be found anywhere along the line will cancel the content of the line preceeding the said character.

III.3.3. Fields of the instruction :

The instruction line can include up to four fields, always in the same order, when present :

a) label field,

b) operation field,

c) operand field,

d) comment field.

UPPER CASE

Fields are separated by one or more blanks. It is therefore very important to note that, except in the comment field or line, the space character may be inserted only to define a field termination.

Each field may contain only a limited number of standard elements, namely :

1) numerical values,

2) symbols,

3) expressions,

4) character spring,

5) special characters.

These different elements will be described in the following paragraphs.

The label field : May only contain a symbol which defines the instruction address. This assembled symbol, therefore, must have a value inferior to 16.383. It may be empty. To do that, the first character of the line must be a space. In this case, the assembler checks the first character of the operation field.

The operation field : contains the instruction mnemonic code as described in paragraph II.2 in the Hardware chapter, or a pseudo-operation as described below in paragraph III.3.5. This field must be present in an instruction line.

The operand field : contains a numerical value, a symbol, an expression or a character spring, according to the operation code of the same line. If the instruction does not require an operand field, it is skipped and the assembler will consider the following field as a comment field.

The comment field : contains any character, excepting the following :

@ which would cancel the line,

Line feed or carriage return , which would terminate the line.

III.3.4. Elements in the fields

III.3.4.1. Numerical values :

A numerical value can be expressed in decimal, or hexadecimal value. They are differentiated by a slash placed in front of the hexadecimal value (diagram 3.1).

356 decimal value 356

/356 hexadecimal value 356

Diagram 3.1.

The range allowed to the numerical value depends upon the context. The diagram 3.2 indicates the range allowed for the numerical values.

Simple numerical operand	Max	/Max
Address operand	255	/FF
Triple precision operand	16383	/3FFF
Output device address	16777216	/FFFFF
Input device address	23	/17
RST operand	7	/7
	7	/7

Diagram 3.2. Maximum value of the numerical data :

In the case of the signed arithmetic, the code used is the two s code, i.e. it will be taken by the assembler as 11111111.
It is very important to refer here to paragraph II.1.3 of the Hardware chapter, indicating that the diagram 2.1 is not valid for a signed arithmetic.

III.3.4.2. Symbols :

The symbol is composed of a maximum of 4 characters. The first must be a "." or a letter, the others alphanumeric characters.

Symbols must be defined either by their presence in the label field or by an EQU pseudo-operation (see paragraph III.3.5.). When it is located in a label field, its value is assigned by the assembler's address counter at the value of the latter at the moment it recognizes the symbol.

There is a certain number of special symbols which will be defined in paragraph III.3.4.5.

```
MARC
.XYZ
A
A1
STOP
LAB
R2E
.126
OFF      (letter O but not zero)
```

Diagram 3.3 Examples of valid symbols

```
MICRAL  To stet long
XY2     "Authorized exclusively at the beginning of the symbol
12A     Figure non authorized at the beginning
A + 3 Non authorized character
A - B  "      "            "
MA RC Blank  in the middle
1-3  Actually, this expression is authorized in the operand field
     exclusively (see next paragraph).
```

Diagram 3.4. Examples of non-valid symbols.

III.3.4.3. Expressions :

An expression is the association of a symbol with a decimal value separated by the sign + or - without blanks.
The symbol must be placed first. The assembler calculates the expression like an arithmetical expression by assigning the value to the symbol as defined by the assembler.

MARC+2	Value of MARC + 2
.18-3	Value of . 18 - 3

Diagram 3.5. Examples of expressions

It is important to avoid any space in an expression because the assembler would interpret it as a field termination.

III.3.4.4. Character spring :

They are composed of characters included between two quotes. Except carriage return, line feed and quote, all the characters are accepted and memorized in the object program in the form of their ASC II code.

III.3.4.5. Special characters :

The special characters accepted by the assembler are the following :

(a) This character cancels the whole lot of characters comprised between itself and the preceeding line and sign.

* Special symbol meaning current address. It may be included in an expression. It cannot be located in the operand field.

< This character may be placed in front of a symbol representing an address. The whole represents the number made of the six highest order bits of the address.

> This character may be placed in front of symbol representing an address. The whole represents the number made of the lowest order bits of the address.

← End of source tape.

' Quote marks delimit litteral character strings.

/ Character indicating that the numeric constant that follows is specified in hexadecimal form.

JMP * -3 Executes a jump at current address - 3

JMP * +3 Executes a jump at current address + 3. Actually, this instruction is equal to NOP, since instruction JMP is a 3 byte instruction and current address + 3 is the instruction that follows JMP.

LLI ADR + 15 loads the low order bits of ADRS increased by 15 units in register L.

3A4 JMP * + Characters of this line are cancelled.

Diagram 3.6 : Example of utilizing special characters.

III.3.5. Pseudo instructions :

The purpose of the pseudo-instructions is to direct the assembler. The list of pseudo-instructions is the following :

ORG
EQU
SET
DC
HDR
END

III.3.5.1. ORG.

Gives the definition of the program origin. The label zone must always be empty, i.e. the first character of an ORG instruction line must be a space.

The operand zone contains a numerical value or a predefined symbol. The value of the operating zone determines the starting address by defining the value of the address counter of the assembler for the first instruction.

III.3.5.2. EQU.

Defines a symbol located in the label zone at a desired value. The label zone must contain the symbol to be defined and the operand zone contains the value assigned to the symbol.

III.3.5.3. SET.

It is a pseudo-instruction equivalent to EQU. The difference between one and the other lies in the fact that EQU defines a symbol for the whole program, while SET defines that symbol up to the point when another SET instruction with same label zone symbol will assign another value.

```
FAC EQU 3   Will assign 3 to FAC in the whole program.
ROM SET 0   It will be loaded with 0 in the second instruction
            and with 1 in the fourth.
LHI ROM 1      dito
ROM SET 1       "
LHI ROM 1       "
```

Figure 3.7 : Example of the use of the pseudo-instructions EQU and SET.

III.3.5.4. DC.

This pseudo-instruction is used in four different forms : DC1, DC2, DC3 and DCK.

Pseudo-instruction DC1 gives the definition of the address of a byte into memory. The label zone contains a symbol which will represent the byte s address in the program and the operand zone contains the byte in numerical form.

PI DC1 3 will write 3 into a memory called PI.

MSKI DCI /OF will write the hexadecimal value /OF into memory called MSKI.

Figure 3.1. Example of the use of pseudo-instruction DC1.

Pseudo-instruction DC2 gives the definition of a number in double precision. The label zone contains a symbol which represent the first byte's address of the value contained in the operand. The first byte contains the low order bits of the number. The high order bits are stored in the following byte.

DEP DC2 / OA15 /15 is located at address DEP and /OA at address DEP+1.

Figure 3.9 : Example of the use of pseudo-instruction DC2.

Pseudo-instruction DC3 gives the definition of the address of a number in triple precision.

Pseudo-instruction DCK gives the definition of the address of a character line. The symbol representing the address must be placed in the label zone and the character line between the two double quote marks in the operand zone. The character line will be placed in consecutive memory addresses under their ASC II code form. The first character is located at the address defined by the label zone.

ASTI DCK "Which channel do you want to select ?"

QSTI address contains /D7, which is the ASC II code of letter W and so on.

Diagram 3.9. : Example of the use of pseudo-instruction DCK.

Figure 3.10 : Instruction DC can be used with a K reproduction factor as indicated in figure 3.10.

DC2K / FF K having a numerical value gives the reproduction of K times /FF, in consecutive memories.

III.3.5.5. HDR

Followed in the operating zone by a character line between double quote marks, permits a change of page, a comment line containing the character line of the operand zone, at page head.

Later on, the page head will appear at each new page until a new HDR pseudo-instruction shows-up.

The label zone must be empty.

III.3.5.6. END

In operation zone without label zone nor operand, shows the assembler that the source tape is finished.

III.4. Assembling

III.4.1. Principle :

Assembling consists of supplying the system with one or several perforated ribbonscontaining the program.

The assembler supplies, upon request :

1°) Diagnosis of error in operation code,

2°) Diagnosis of error in addressing,

3°) A perforated ribbon containing the object program,

4°) Complete listing containing the object related to the source codes and the commentaries,

5°) A cross table of references.

III.4.2. Loading of the assembler :

If the assembler is not placed in ROM, it must be loaded from the monitor and executed from its starting address.

If the assembler is placed in ROM, it will be possible to go into in stop mode and substitution mode starting from the console by putting in instruction JMP at the assembler's input point three times, thereafter by reverting to automatic mode.

III.4.3. Loading of source program

As soon as the assembling program is in the execution phase, it sets the processor in Wait state waiting for the introduction of one or several perforated ribbons containing the source program.

Each ribbon contains an end-ribbon sign which is taken into account by the assembler.

As taking instructions into account proceeds, the assembler points out errors in the operation code through diagnosis as shown on diagram 3.11.

When pseudo-instruction END appears, assembling starts and gives possible diagnosis of address error as shown in diagram 3.11.

When assembling is over, a binary ribbon representing the object program is generated by the teletype's perforator.

A dialogue takes place between the operator and the system. It allows the operator to choose either of the following solutions :

1°) New start at the assembler's source to assemble a new program;

2°) New perforation phase of the binary program. The operator can request perforation of the complete program;

3°) Generation of a complete listing. In such cases, it is necessary to present the symbolic program once again, ribbon by ribbon;

4°) Generation of the cross reference.

This cross reference gives the list of the symbols used with their definition and utilization address so as to make any corrections easier.

ANNEXES

A1	0. VOLT
A2	AINT1/
A3	AINT6/
A4	AINT2/
A5	
A6	MD1/
A7	AINT3/
A8	ϕ2/
A9	AINT5/
A10	BI2/
A11	S4/
A12	BI5/
A13	BI4/
A14	BI6/
A15	D2/
A16	S2/
A17	D1/
A18	D3/
A19	BI7/
A20	S7/
A21	S6/
A22	S5/
A23	MD5/
A24	D5/
A25	MD6/
A26	S9/
A27	T2/
A28	T'3/
A29	D6/
A30	CC0/
A31	CC1/
A32	MD3/
A33	S11/
A34	MD0/
A35	T3/
A36	STOP
A37	0. VOLT

B1	+5V
B2	-9V
B3	BI1/
B4	AINT7/
B5	
B6	READY C
B7	AINT4/
B8	RZGI/
B9	READY/
B10	BI3/
B11	SUB/
B12	
B13	
B14	
B15	S3/
B16	S0/
B17	S1/
B18	
B19	MD4
B20	
B21	
B22	
B23	
B24	D4/
B25	MD7/
B26	S8/
B27	D0/
B28	INIT/
B29	MD2/
B30	D7/
B31	S13/
B32	SYNC/
B33	S10/
B34	S12/
B35	PROTECTION OF
B36	RAM MEMORY
B37	+5V

ANNEXE 1 - PLURIBUS INTERCONNEXION

ANNEXE 2 : TABLEAU DES ETATS MACHINE

CHARGEMENTS

Code Machine $D_7D_6D_5D_4D_3D_2D_1D_0$	Opération	NOMBRE ETATS	1er Cycle Mémoire : O P T1	T2	T3	T4(3)	T5	2ème Cycle Mémoire	T1	T2	T3	T4	T5	3ème Cycle Mémoire T1	T2	T3	T4	T5
I I D D D S S S	Lr1r2	(5)				S S S Lu	D D D Chargé											
I I D D D I I I	LrM	(8)	(1)			7 →		LEC	(2)									
I I I I I S S S	LMr	(7)	(1)			S S S LU	→	LEC	(2)									
O O D D D I I O	Lrl	(8)	(1)			→		ECR	(1)									
O O I I I I I O	LM1	(9)	(1)			→		LEC	(1)					ECR L → S0/S7/	H → S8/S13			

OPERATIONS ARITHMETIQUES ET LOGIQUES

O O D D D O O O	1Nr	(5)		x	Indicateurs positionnés												
O O D D D O O I	DCr	(5)	(1)	x	Indicateurs positionnés												
I O O P P P S S S	OPr	(5)	(1)														
I O P P P I I I	OPM	(8)		→		LEC	(2)						Opération Indicateurs					
O O P P P I O O	OPI	(8)		→		LEC	(1)						Opération Indicateurs					
O O O O O O I O	RLC	(5)		x	Rotation												
O O O O I O I O	RRC	(5)		x	Rotation												
O O O I O O I O	RAL	(5)		x	Rotation												
O O O I I O I O	RAR	(5)		x	Rotation												

RUPTURE DE SEQUENCE

O I X X X I O O	JMP	(II)		→		LEC	(2)			→	LEC	>C0→S0S7/	C0→S8/S13/	Prise	Charge	Charge	
O I O C C O O O	JFc	(9 or II)	(1)	→		LEC	"			→	LEC	"	"	en	Partie	Partie	
O I I C C O O O	JTc	(9 or II)		→		LEC	"			→	LEC	"	"	compte	haute	basse	
O I X X X I I O	CAL	(II)		→		LEC	"			→	LEC	"	"	adresse	compteur	compteur	
O I O C C O I O	CFc	(9 or II)		→		LEC	"			→	LEC	"	"	haute	ordinal	ordinal	
O I I C C O I O	CTc	(9 or II)		→		LEC	"			→	LEC	"	"				
O O X X X I I I	RET	(5)		Pile affectée	x												
O O O C C O I I	RFc	(3 or 5)		Pile affectée	x												
O O I C C O I I	TRc	(3 or 5)		Pile affectée													
O O A A A I O I	RST	(5)	En plus, en T3 pile affectée	O dans C.O	Adresse dans C.O.												

ENTREES SORTIES

O I O O M M M I	INP	(8)		→		E/S	(3)			(3)							
O I R R M M M I	OUT	(6)		→		E/S	(4)										

INSTRUCTIONS MACHINE

I I O I O O I O	MAS	(5)		Exécution	Exécution												
I I I I O I I O	DMS	(5)	(1)	"	"												
O O O I I I I I	REI	(5)	(1)	Exécution													
O I I I I I I I	OUT/17	(5)				E/S	(3)										
O O O O O O O O	HLT	(4)															
I I I I I I I I	HLT	(4)															
I I O O O O O O	NOP	(5)		Exécution	Exécution												

(1) en T1 partie basse du compteur ordinal dans le registre d'adresse S0/ à S7/ ; en T2 idem partie haute dans S8/ à S13/ ; en T3 prise en compte de l'instruction
(2) en T1 Registre L dans registre d'adresse (S0/ à S7/); en T3 Registre M dans S8/ à S13/ ; en T3 prise en compte de l'instruction.
(3) en T1 A dans registre d'adresse (S0/ à S7/) ; en T2 instruction dans S8/ à S13/ ; en T3 entrée des données ; en T4 indicateurs sur BUS (B0 = S, B1 = Z, B2 = P, B3= C)
(4) en T1 A dans registre d'adresse (S0/ à S7) ; en T2 instruction dans S8/ à S13/ ; en T3 inactif ; en T5 les données dans le registre A

APPENDIX 3

Detailed operating of the processing card :

The processor starts by delivering the low address of the ordinal counter which is memorized in T1, afterwards it delivers the high address, memorized in T2 simultaneously with bits CCO and CC1 which are both at the low level, indicating that the processor is in an OP cycle which searches the memory for the instruction to be executed.

When the memory is ready, it sends a low voltage current into the READY terminal of the pluribus. This instruction must arrive during sate T3 (+).
If the memory thus used is too slow (cycle addressage reading superior to 4 µs), it is necessary to send a high level into terminal READY to put the processor on stand-by until the memory has delivered the instruction. READY can then be taken down to a low level, and the processor will deliver T3 which commands decoding of the instruction, its executing taking place in T4 and/or T5 when it includes only one cycle. If it includes more than one memory cycle, cycles T1, T2, T3, T4 and T5 re-appear. They all differ depending on the instruction.

Board Ann. 2 shows the general sequence of all instructions.
In this board, the three first stata of each instruction are jumped because they are identical :

T1 low part of the output ordinal counter,

T2 high part output,

CCO and CC1 positioned at OO

T3 instruction accounted for and decoded.

Examples :

Instructions CFZ : it is indexed in the board as CFc.

- the first column shows that binary code is 01001010.
- the third column calculates its execution time in terms of basic cycles, i.e. 9 cycles when the condition is not realized (CAL non-executed) and 11 cycles when it is realized.

(+) Actually, the processor takes state T'3 into account during an auxiliary which is elaborated to indicate all exchanges with the processor.

The following columns indicate the content of the states given below :

First memory cycle :

T1 : the low part of the ordinal counter is memorized in the output register and is available in SO/ to S7/ in the pluribus.

T2 : the high part of the address is memorized in the output register and available in S8 to 13 in the pluribus (CCO and CC1 are positioned at OO (PCI).

T3 : the instruction is accounted for on DO to D7 in the pluribus.

T4 and T5 are leaped over.

Second memory cycle :

T1 and T2 : similar states to those of the first external cycle, except CCO and CC1, which are respectively positionel at O and 1, indicating that the processor is in the memory data reading state (PCR).

T3 : the low address of any leap is taken into account and memorized into an inaccessible register.

T4 and T5 are leaped over.

Third memory cycle :

T1 and T2 : identical state as those of second cycle.

T3 : the high address of any leap is taken into account and memorized into a second inaccessible register; indicator are tested and switching is ordered.

T4 : when the condition is not reached, T4 is leaped over.
When it is, the stack's pointer is advanced and the content of the second inaccessible register and loaded into the high part of the ordinal counter.

T5 : (leaped of when the condition is not reached). The contents

APPENDIX 4

Operating of programmed logics.

This appendix is meant for users who are not familiar with computers and who want to use MICRAL to replace cable logics.

Structure of a computer :

MEMORIES

CENTRAL UNIT

CONNECTION WITH EXTERNALS

From a philosophical view-point, a computer can be taken as made of three parts :

1) the memory which contains the programs and the data giving control of the desired automatism;

2) the processor which controls the automation by executing the instructions contained in the program;

3) connection with externals tying the computer to the system.

The connection with surrounding is made of two parts :

- inputs-outputs which permit the information to flow between the computer and the externals;

- the interruption system which permits the externals to reach the computer giving priority to certain programs by needing development by the process.

The memory :

The memory has two functions :

1) stockpile the programs to deliver them to the processor for execution;

2) stockpile the data needed to execute the program, from parameters known in advance by the programmer and memorized as programs, or from externals and memorized during processing after any processing by the processor.

As a consequence, there are two types of memories :

1) memories that contain the programs and the parameters known in advance and that can be memorized upon conception of the system. For that, Read-Only memories can be used, since it is not necessary to write them during process;

2) memories that contain data supplier by the process and therefore must be able to be written or read.

   Random-Access memories must be used to for this (see paragraph especially describing memories).

Organization of the memories

To utilize a memory, one must first identify the compartment to be used. That is done by the address register and decoder. Each time that access to the memory is wanted, first load the address register. The decoder will then select the memory and thereupon it will be possible to read the selected memory or modify its content.
From the programmer's viewpoint, the main memory of MICRAL is divided into pages. Each page contains 256 bytes (eight bit word). The standard memory can contain 64 pages equal to 16 K words of memory.

Contents of memories

As seen above, memories contain eight-bit binary words representing either data or instructions, depending on the context. The data, as well as the instructions, will have to be interpreted to be of use. This is the processor's function.

Numerical data

If the data represents a numerical value, it will be generally kept in the binary form as to save maximum space within the memory. A byte may contain a figure ranging only between 0 and 255. When a larger dynamic is required, several bytes must be used to represent the number. The central unit will then process the number, byte by byte. In that case, the processor is said to work in multiple precision (when working with two bytes, it is said to work in double length).

When a negative number is to be represented, ✳ is utilized.

To understand the concept of ✳ , imagine a counter with a capacity of four digits whose initial value would be 0001. Deduct 1, leaving 0. Had 1 been deducted, the result would be 9999, while the desired result is - 1.

A similar concept is used to represent negative binary numbers as to simplify adding and substracting.

The ✳ of a number A is defined as the number which, added to A, results into zeros at all weights of the binary number and results into a carry-over beyond the highest order bit.

A 101 000 11 Diagram 2 shows the ✳ B of a A byte.

BB 01 01 11 01

Diagram 2

The ✳ is obtained by completing all the bits of the byte and by adding arithmetically the number 1 (diagram 3).

A

plus 1

B

Diagram 3

gives the number sign.

If it is 1, the number is negative.

If it is 0, the number is positive.

In a multiple precision number, this applies only to the highest order bit (where the sign is concerned) and to the lowest order bit (where number 1 is concerned) to obtain the ✳ and one must operate the carry-overs byte by byte.

Non numerical data

Non numerical data is interpreted by the processor and by the input/output peripherals.

For example, the alphanumerical characters can be memorized in the form of their ASC II code.
Likewise, state words can be memorized and analyzed bit by bit (each bit bearing a symbol supplied by the context) by rotating the word inside the central unit. The bit's value can be determined either by testing the contents of the carry, or by testing the sign of the number which would represent the word in its present position (see preceeding paragraph for the carry and the sign, as well as diagram 3).

	CARRY	SIGN	WORD
before rotation			
1st			
2nd			
3rd			

AV : the carry indicates nothing, the sign shows that D7 = 1

(1) the carry indicates that D7 = 1 and the sign that D6 = 0

(2) the carry indicates that D6 = 0 and the sign that D5 = 1

(3) the carry indicates that D5 = 1 and the sign that D4 = 0

Diagram 3 : analysis of a bit by rotation to the left.

The example as per diagram 3 shows that the sign test is faster than the CARRY test. However, the CARRY test permits to shift the analysed bit.

PROGRAMS

The memory also contains the instructions making up the program. Each instruction is called byte by byte into the central unit to be interpreted (decoded) and executed.

The central unit is made in such a way as to allow each configuration of the word to correspond to an action that the processor will have to execute.

For instance, in MICRAL, the word 00 000 010 means rotation of a word placed in a register called accumulator (this instruction enables the analysis of the bits of the word contained in the accumulator, as seen above.

So, a byte could control the central unit in 256 different ways which the programmer must know to be able to prepare his program. In order to make the programming work easier, each instruction is associated with a mnemonic code.
For instance, RAL (Rotate Accumulator Left) will be the code associated with 00 000 010.

Development of the program :

The memory's addressing register is loaded automatically by the processor from a register called the ordinal counter. In the normal mode, instructions are called in sequence. To achieve that, the processor advances the ordinal counter at each instruction, but certain instructions cut-off the sequence by loading the ordinal counter with a different value, thus allowing the use of modifiers.

There are two sort of switches :

- those foreseen by the program,
- those generated by the system.

The later are called interruptions. Interruptions can have different priority levels. It means that an interruption can itself be interrupted, provided it is done by an interruption having a higher priority level. The process has the ability to set itself free from interruptions by means of the program.

It can :

1° it can mask all the interruptions. They all become then inoperative. In MICRAL, it is impossible to mask the zero level interruption (catastrophic);

2° or by disenabling them individually.

The programmed indicators can be of two types : unconditional or conditional. Conditional indicators consist of testing the processor and of connecting the program either in sequence or on a new specific address according to the state of the indicator.

In MICRAL, there are three types of indicators :

1° the leaps consist of simply switching the program;

2° the sub-program calls allow, in addition, the memorizing of the ordinal counter's address at the time of the leap;

3° the sub-program returns which allow connection to the address memorized when the call takes place.

imp. TECHNIPLAN 805 70 81